Jumpshot-4 Users Guide

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

July 11, 2007

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

Acknowledgments

We 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 U.S. 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

Chapter 1

Introduction

Jumpshot-4 is a visualization program for the logfile format, SLOG-2, which provides a hierarchical structure to store a large number of drawable objects in a scalable and efficient way for visualization. SLOG-2's new scalable logfile format allows the display program to provide functionalities never before possible. Level-of-detail support through preview drawables provides high-level abstraction of the details without reading huge amounts of data into the graphical display engine. Jumpshot-4 allows seamless scrolling from the beginning to the end of the logfile at any zoom level. In addition, new functionalities are available, such as dragged-zoom, grasp and scroll, instant zoom in/out, easy vertical expansion of timelines, and cut and paste of timelines. A new search-and-scan facility is provided in order to locate hard-to-find objects in a very large logfile. Also, the histogram module based on user-selected duration provides a convenient and graphical way to analyze the statistics of a logfile (e.g., it enables 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, including CLOG, CLOG-2, RLOG, and UTE, and it conforms to the standard look and feel 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 the 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 graphical objects contained in the SLOG-2 file are drawn on the canvas. These objects are called *drawables.* There are two kinds of drawable objects:*primitive* and *composite* drawables. The primitive drawables are the simplest drawables and are considered to be basic elements of the SLOG-2 file. They are categorized based on their topological structures. Currently, three topologies are supported in SLOG-2:*state, arrow,* and *event.* Both state and arrow are drawables identified by two points in the timeline canvas, that is, a pair of (timestamp, timeline ID) coordinates. State's start timeline ID is the same as its final timeline ID, but arrow's start and final timeline IDs may be different. Event consists of only one point in the timeline canvas; that is, it has only one timestamp and one timeline ID. The 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 drawableare 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; that is, the nested states correspond to the nested subroutine calls. The current implementation of the SLOG-2 format stores some of the state nesting information to optimize the performance of the visualization program.

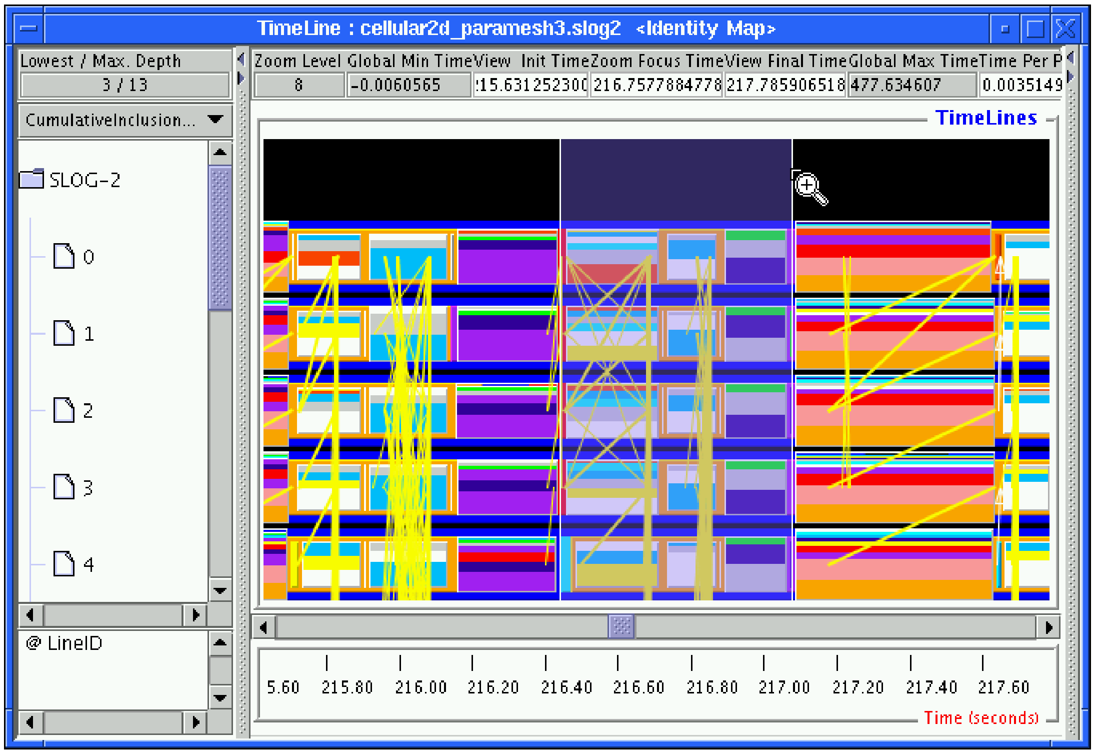
2.2 Understanding the Preview Drawable

The preview drawable is created as a result of renormalization 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. The preview drawable is designed to amalgamate real drawables of the same topological type, for example, a preview state is a "state amalgamates only" object.

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

Hence, a preview drawable is always a primitive drawable in the renormalization scheme. There are currently three 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 can appear in the Legend window of the display program (see Figure 3.6. The Legend window contains a table of legends that are basically a visual representation of the 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 the preview drawable to its detailed content of the first five processes of a 16-process MPI slog2 file when zooming in on the timeline canvas. The sequence of figures is generated by zooming in on a marked region in each successive figure in sequence. The marked region is shaded and is bounded by a pair of white lines. A magnifying glass with a plus sign in the center is the cursor that marks the ends of the zoom region. Figure 2.1 is a typical timeline canvas, in which most of the real drawables are still buried inside their preview drawables. In the figure are preview arrows, preview states in the front, and some long-running real states in the back.


TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per P
3 / 13	8	-0.0060565	!15.631252300	216.7577884778	217.785906518	477.634607	0.0035149

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

Each thick yellow line is a *preview arrow,* which represents a collection of arrows between its two ending timelines. The start and final timestamps of the 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 time; it indicates simply that there are arrows starting or ending within these two times and between the two 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 a preview arrow can have, the thickness of the preview object is set equal to the order of magnitude of the number of real objects amalgamated. That is, the same thickness in two different preview arrows does not mean that they contain exactly the same number of real arrows; rather, it means that the numbers of real arrows contained in the preview objects are within the same order of magnitude, that is, within a constant multiplicative factor as defined by PREVIEW_ARROW_LOG_BASE in the Preference window shown in Figure 3.28 and in Table 3.20. 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 the *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 the left side of the y-axis label,[2] the distribution and the heights of the strips can be changed dramatically. One of the display options for the preview state is *CumulativeInclusionRatio*. With this option, the strips are arranged in order of decreasing height (somewhat 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,*that is, disregarding the nesting state order. This visual representation tells which state categories can be within the span of the preview state and which state category contributes the most statistically to the specified time range, so that the user can decide where to zoom in to find out more details. In a sense, the preview states provide a global, coarse-grained summary of what is going on, without losing as many details as with the preview in the older version of Jumpshot. For example, the new preview states retain timeline ID information, which may enable early detection of load-balancing problems before zooming in to see all the real states.

Figure 2.2 shows a zoomed-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 have been 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 the preview state becomes apparent in the figures: Preview states are properly nested within real states. In the most expanded y-axis label view, the preview state is always on top of the other nested states;[3]that is, states that enclose the preview state are always 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 configuration indicates that the white, turquoise, and gray real states are all nested inside the long-running orange and dark royal-blue states.

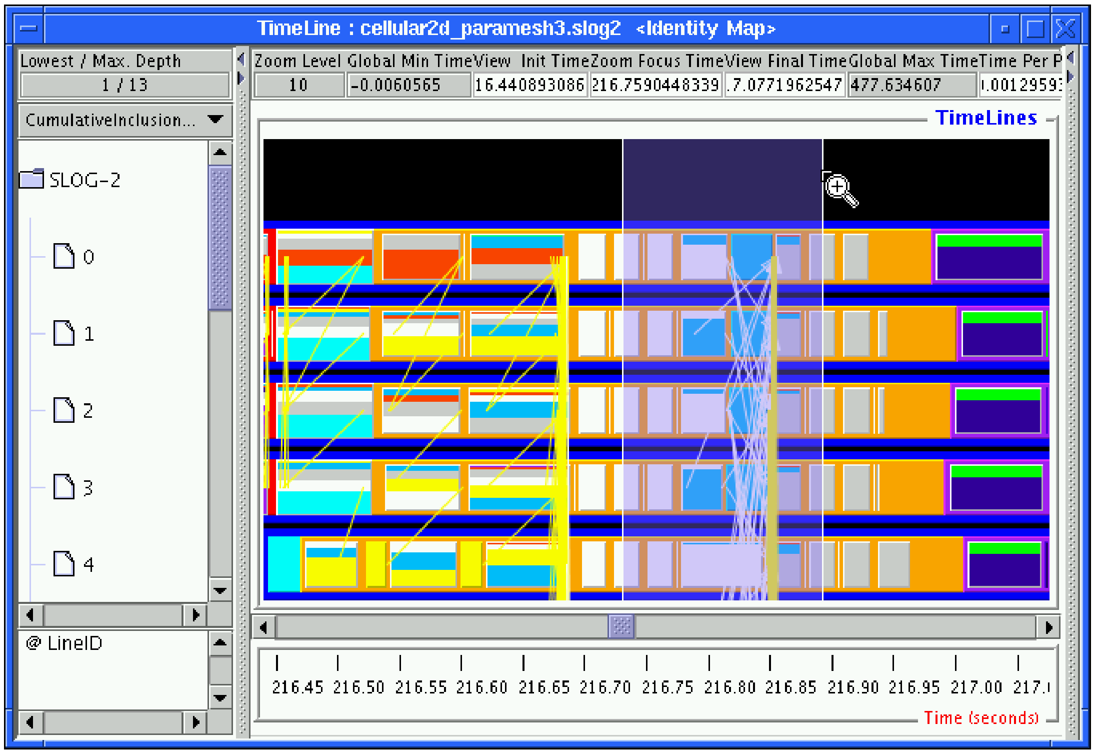
Figure 2.3 is a zoomed-in view of the region marked by the pair of white lines in Figure 2.2. Comparing these two figures, we see that all the preview drawables have been replaced by real drawables. Each white preview state is replaced by hundreds of white real states. The same is true for the gray preview states to the rightof the turquoise states.[5] The preview arrows all have been replaced by

[2]In the Preference window, as shown in Figure 3.28 and in Table 3.20, there is also a PREVIEW_STATE_DISPLAY variable. The variable determines the initial PREVIEW_STATE_DISPLAY used when the Timeline window is first made visible.

[3]Only in a slog2 file that has multiple ViewMaps and where timelines can be collapsed, that is, AIX's UTE generated slog2 file, can a preview state be nested with other preview states in a 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.

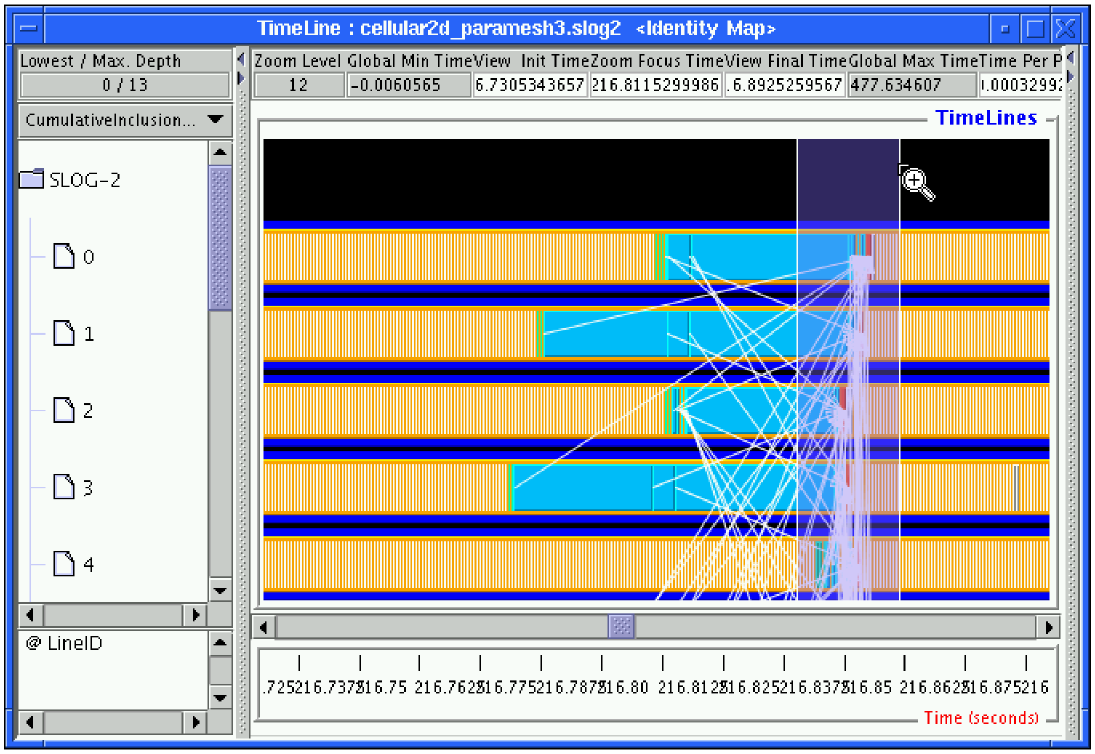
[5]In order to speed the graphics performance of the display program, an aggressive algorithm has been used to eliminate drawing states that are closely packed together within the nearest neighboring pixels. Together with the fact


TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth
1 / 13

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per P
10	-0.0060565	16.440893086	216.7590448339	.7.0771962547	477.634607	.00129593

Figure 2.2: Zoomed-in view of Figure 2.1.

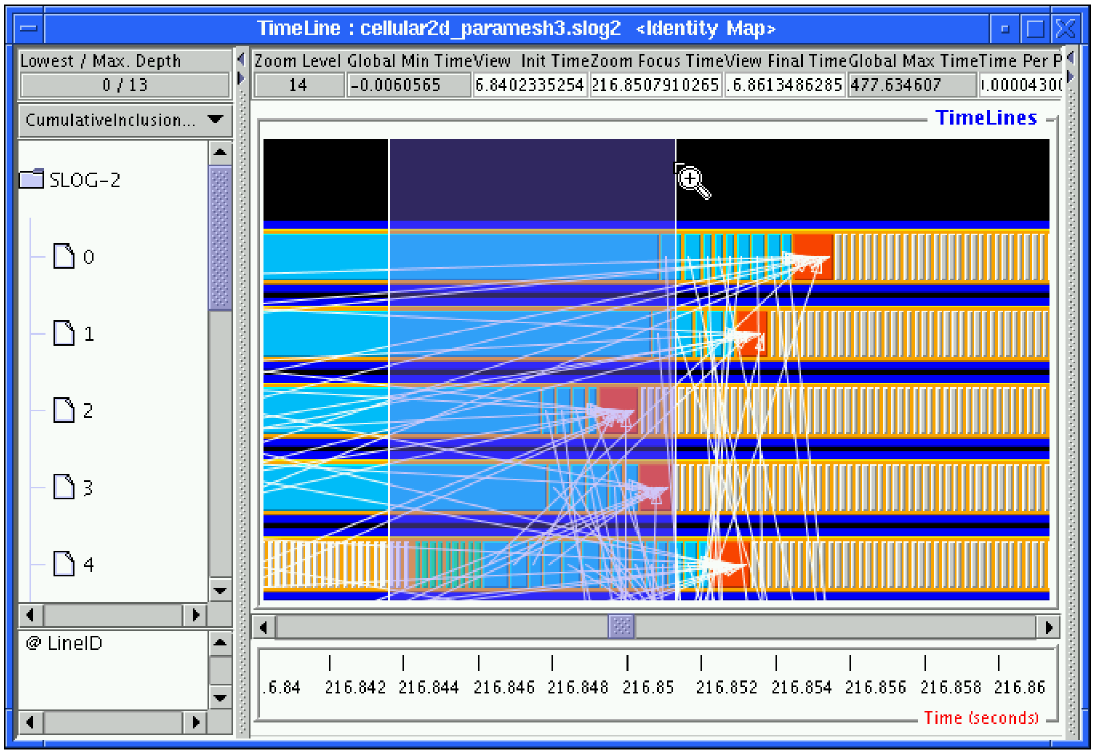

TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth
0 / 13

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per P
12	-0.0060565	6.7305343657	216.8115299986	.6.8925259567	477.634607	.00032992

Figure 2.3: Zoomed-in view of Figure 2.2.

real arrows. The region marked by the white lines 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-in region. This can be achieved by right clicking on the preview drawable. The result is shown in Figure 3.17.


TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth
0 / 13

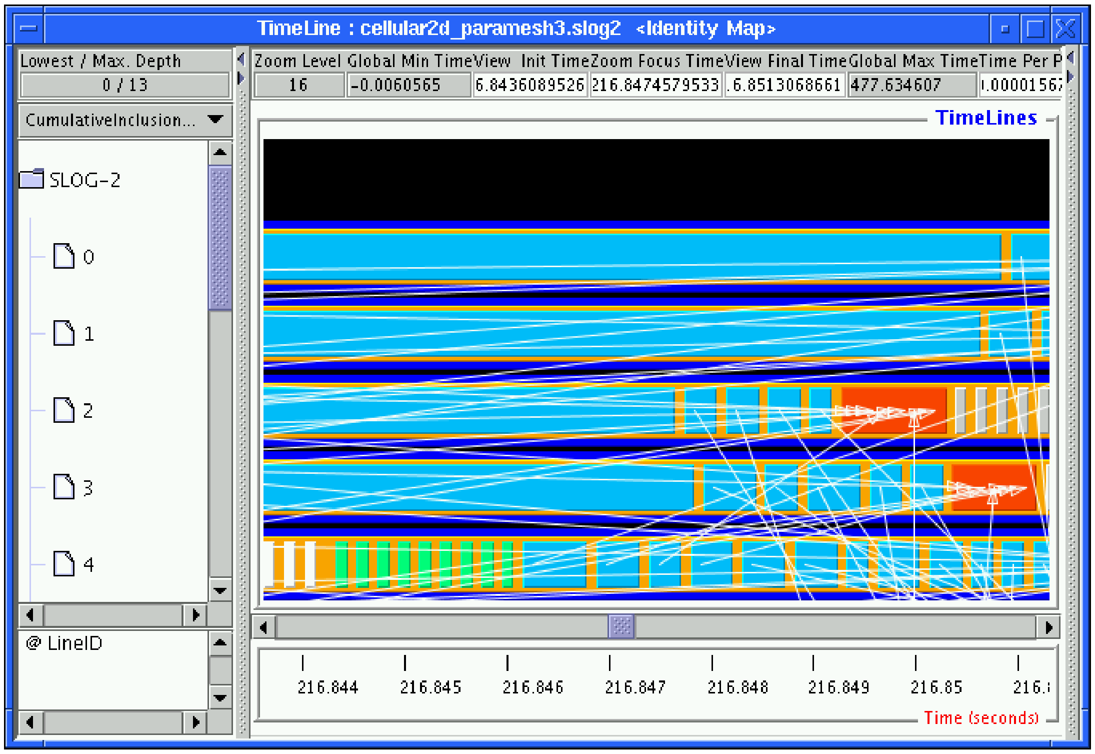
CumulativeInclusion...

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per P
14	-0.0060565	6.8402335254	216.8507910265	6.8613486285	477.634607	.00004300

Figure 2.4: Zoomed-in view of Figure 2.3.

Further zooming in on the region marked by the white lines 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.

that the number of pixels available is less than the number of nonoverlap states in the region, the number of the real states may sometimes not appear as numerous as the Drawable Info Box of the preview state indicates. In that case, a further zoom-in will be needed to confirm the case, as shown in Fig. 2.4.


TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per P
0 / 13	16	-0.0060565	6.8436089526	216.8474579533	.6.8513068661	477.634607	ι.0001567

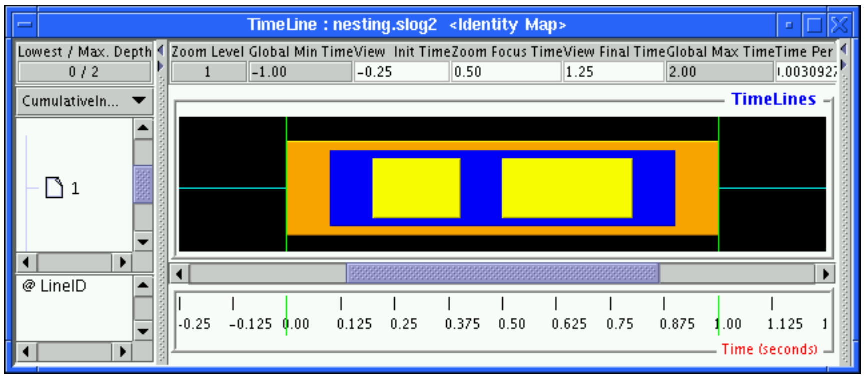
CumulativeInclusion...

Figure 2.5: Zoomed-in view of Figure 2.4.

2.2.1 Understanding the Preview State Display

So far only one of the representations of the preview state, *CumulativeInclusionRatio*, has been used to illustrate the concept and representation of the preview state. Jumpshot-4 actually uses several different representations of the preview state. All these representations are based on two ratios stored in the SLOG-2 file: *inclusion ratio* and *exclusion ratio*[6]. The inclusion ratio is computed without taking into account the nesting order of the states. States that either are nested inside or enclose other states contribute equally to the inclusion ratio. The 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, the 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 for computing these two ratios is to satisfy two opposite needs of the preview state. The MPI application developer who has put a lot of user-defined states in a SLOG-2 file, through either MPE or AIX's PCT utility, is likely to be interested in the profiling information of the user-defined states that enclose MPI states and other user-defined states. In this case, the inclusion ratio will be useful. The 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 a high preview level. On the other hand, the MPI implementor or the person interested in the low-level MPI networking overhead is likely to be interested in the profiling information of MPI and its internal calls. The exclusion ratio will come in handy here. 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 a very high preview level.


TimeLine : nesting.slog2 <Identity Map>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per
0 / 2	1	-1.00	-0.25	0.50	1.25	2.00	ı.0030927

Figure 2.6: 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, and the sum of durations for the two 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 turn nested in the orange state. The pair of green lines mark the region where a preview state is being created.

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

[6]The exclusion ratio computed in SLOG-2 is less than or equal to what it should be. This artifact is due to the fact that preview state is used in the determination of exclusion region. The nesting level of preview state is approximate by construction. This approximate nature of the preview state may exclude more region in the enclosing state than what the appropriate shares of its enclosed states should be. Nevertheless, even with this limitation, the innermost state's exclusion ratio is still correct.

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: Contributions of real states to a preview state of duration 1.0 sec as marked by the pair of green lines in Figure 2.6.

the orange outermost state, but the most dominant 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.

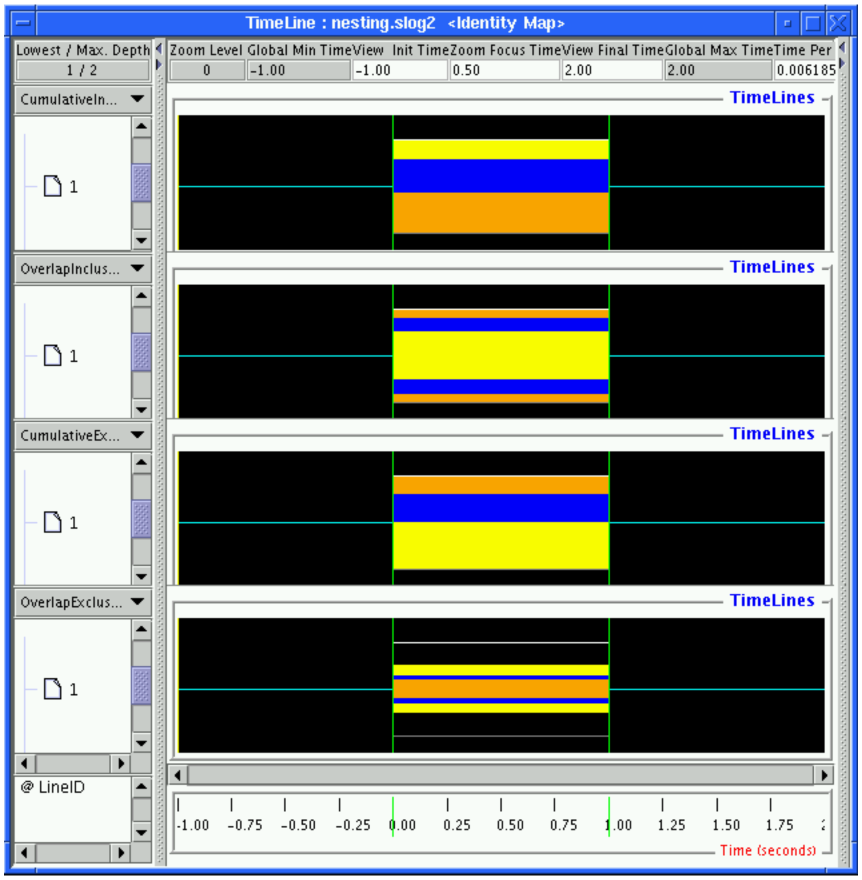
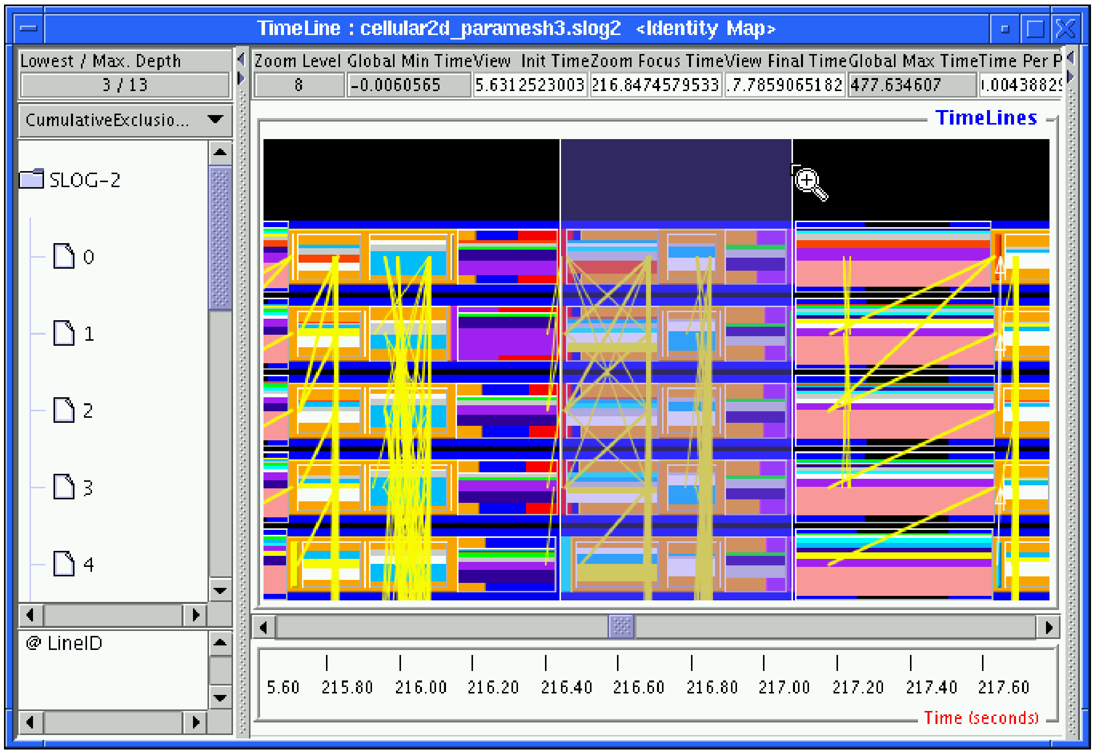


Figure 2.7: Different preview state displays of the zoomed-in view of the Figure 2.6. Starting from the top, the first one is the *CumulativeInclusionRatio* view, the second one is the *OverlapInclusionRatio* view, the third one is the *CumulativeExclusionRatio* view, and the last one is the *OverlapExclusionRatio*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

With the 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. This statement is true for all preview state displays except *CumulativeInclusionRatio,* which may have its total sum of ratios in excess 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 the least important in the top *CumulativeInclusionRatio* view but becomes the 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, the *CumulativeInclusionRatio* view cannot be used to compare different preview states because of the arbitrary rescaling.[7] If one is interested in comparing inclusion ratios across different preview states, the *OverlapInclusionRatio* view can be used instead. This view draws all inclusion ratios proportional to the height of the preview state but in an overlapping way, that is, in order of decreasing inclusion ratios, and stacks one on top of the other (somewhat like a nested state). The overlap view of exclusion ratios is the *OverlapExclusionRatio* view, shown at the bottom of Figure 2.7. The *OverlapExclusionRatio* view draws exclusion ratios exactly the same way as does the *OverlapInclusionRatio.* In general, an overlap view cannot fill up the full height of the preview state. This is apparent in the *OverlapExclusionRatio* view in Figure 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. The white bordered box can sometimes be confusing, however, because whatever is 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 the PREVIEW_STATE_BORDER in the 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 provides an extra dimension of information compared with its inclusion ratio counterpart, at the expense of being a bit more complicated visually.

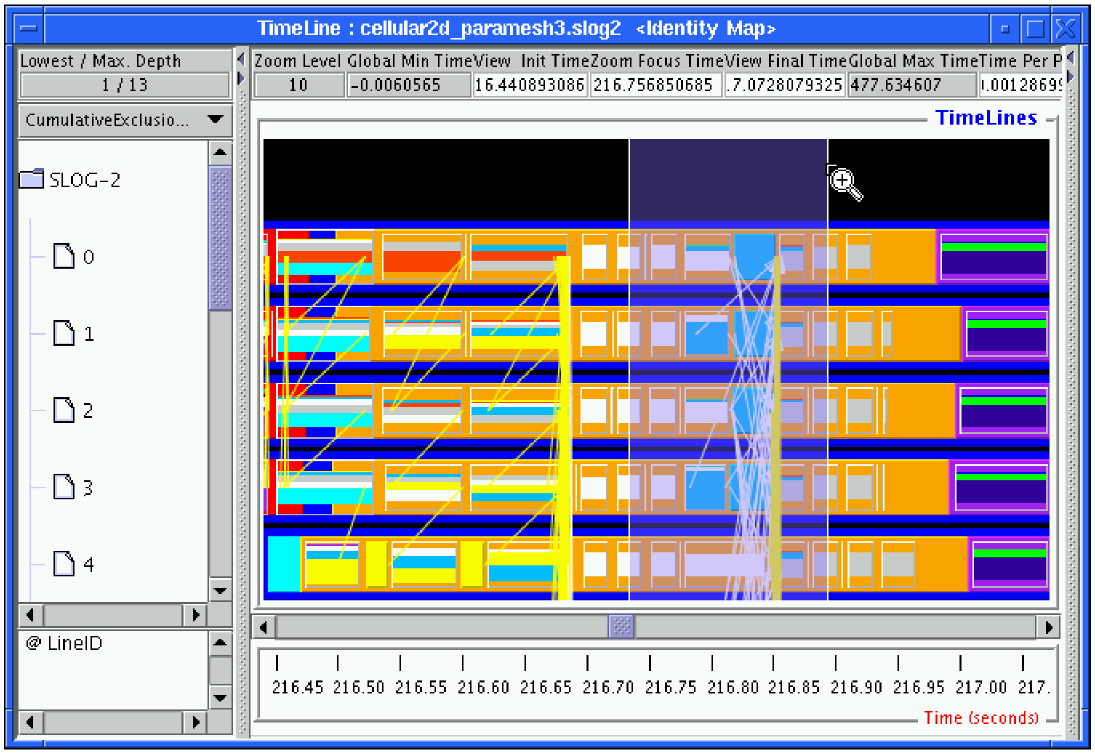
[7]Usually, neighoring preview states in the *CumulativeInclusionRatio* view have a similar total sum of inclusion ratios. Hence, one can compare adjacent preview states. But we note that the total sum of inclusion ratios between nearby preview states can change dramatically without any visual indication. When in doubt, one should right click on the preview state to get the Drawable Info Box and confirm the ratios.


TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth: 3 / 13

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per P
8	-0.0060565	5.6312523003	216.8474579533	.7.7859065182	477.634607	ı.00438825

Figure 2.8: *CumulativeExclusionRatio* view of Figure 2.1.


TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth: 1 / 13

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per P
10	-0.0060565	16.440893086	216.756850685	.7.0728079325	477.634607	ı.00128699

Figure 2.9: *CumulativeExclusionRatio* view of Figure 2.2; also, a zoomed-in shot of Figure 2.8.

Chapter 3

Graphical User Interface

3.1 Main Window

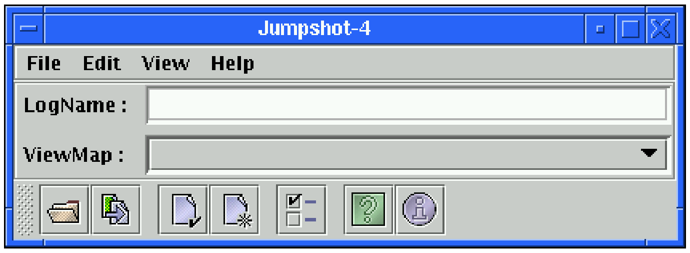


Figure 3.1: Main control window of Jumpshot-4.

The first window that pops up when invoking Jumpshot-4 is called the Main window, as shown in Figure 3.1. The buttons shown in the toolbar are shortcuts to the submenu items in the top menu bar. The function of each of these buttons is listed in Table 3.2. Two text fields display crucial information about the logfile being processed. The text field entitled *LogName* displays the pathname of the logfile being processed. The pulldown menu entitled *ViewMap* lists all the available ViewMaps in the SLOG-2 file. The CLOG[1], older CLOG-2[2] and RLOG-converted[3] SLOG-2 files contain one ViewMap, called the Identity Map. The recent CLOG-2 and IBM's UTE trace-converted SLOG-2 file contains multiple ViewMaps, e.g. a CLOG-2 logfile generated from a multi-threaded MPI program contains the Process-Thread and Communicator-Thread ViewMaps besides the Identity Map.

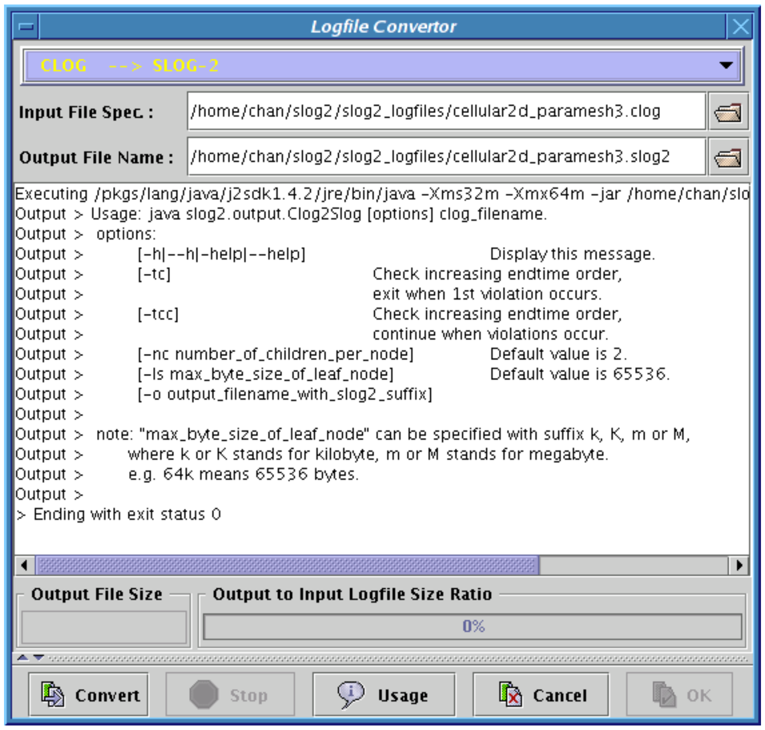
3.2 Logfile Convertor Window

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. Currently, five convertors are supported: *CLOG –> SLOG-2*, *CLOG-2 –> SLOG-2, RLOG –> SLOG-2, UTE –> SLOG-2* and *TXT –> SLOG-2.* The convertor is generally selected based on the input file's file extension. If the wrong file convertor is selected, the user can correct it through the pale-blue

[1]A low-overhead native trace format from MPE.
[2]A low-overheaed native trace foramt from MPE-2
[3]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 Users Manual	show the Users Manual of this program
ℹ	Show FAQs	show the FAQs of this program

Table 3.2: Functions of the toolbar buttons



Input File Spec. : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.clog

Output File Name : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.slog2

```
Executing /pkgs/lang/java/j2sdk1.4.2/jre/bin/java -Xms32m -Xmx64m -jar /home/chan/slo
Output > Usage: java slog2.output.Clog2Slog [options] clog_filename.
Output >  options:
Output >       [-h|--h|-help|--help]                      Display this message.
Output >       [-tc]                              Check increasing endtime order,
Output >                                          exit when 1st violation occurs.
Output >       [-tcc]                             Check increasing endtime order,
Output >                                          continue when violations occur.
Output >       [-nc number_of_children_per_node]          Default value is 2.
Output >       [-ls max_byte_size_of_leaf_node]           Default value is 65536.
Output >       [-o output_filename_with_slog2_suffix]
Output >
Output >  note: "max_byte_size_of_leaf_node" can be specified with suffix k, K, m or M,
Output >      where k or K stands for kilobyte, m or M stands for megabyte.
Output >      e.g. 64k means 65536 bytes.
Output >
> Ending with exit status 0
```

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

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 an update of the Output File Name field with the default name. The Logfile Convertor has five major functions, each is associated with a button in the lower panel of the window. They are listed in Table 3.4.

Icon	Description	Function
➡	Convert	Start the logfile conversion of the selected convertor
🛑	Stop	Stop the ongoing logfile conversion of the selected convertor
ℹ	Usage	Print the usage information of the selected convertor
✗	Cancel	Close the window without doing anything
✓	OK	Display the last converted SLOG-2 file and close the window

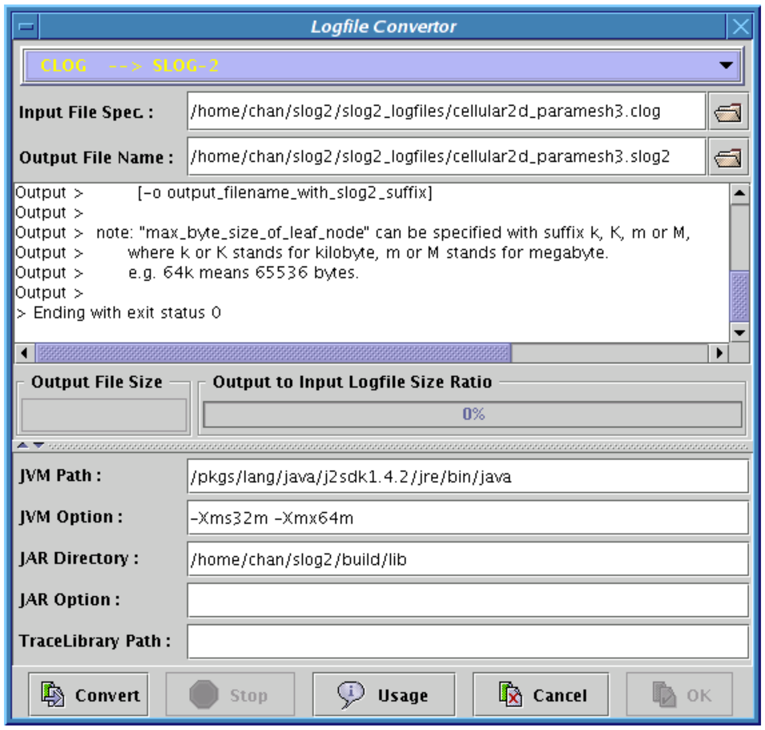
Table 3.4: Major functions in the Logfile Convertor window.

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 needed by any logfile convertor are supplied through a panel hidden by a splitter in the convertor window. The splitter has a divider that can be lifted up to display all the parameters used to launch the Java process, as in Figure 3.4. On the rare occasion that the default parameters are not correct, the text fields can be modified to reflect the situation.

The logfile conversion process is started by hitting the *Convert* button. The standard output and error streams of the process are 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 Figure 3.4. During the conversion, only the *Stop* button is enabled for the case that user wants to stop the ongoing conversion.

If the logfile conversion fails, the error message will be printed in the text area for diagnosis or a bug report. As shown in Figure 3.5, the *OK* button is enabled only when the logfile conversion is terminated normally and the *STOP* button has not been clicked during the conversion.

If *OK* button is clicked, the last converted slog2 file will be used for the subsequent visualization. If *Cancel* button is clicked, the Logfile Convertor dialog will be closed and the control is returned to the Main window as if the Convertor dialog has never been invoked.



Input File Spec. : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.clog

Output File Name : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.slog2

```
Output >         [-o output_filename_with_slog2_suffix]
Output >
Output >  note: "max_byte_size_of_leaf_node" can be specified with suffix k, K, m or M,
Output >        where k or K stands for kilobyte, m or M stands for megabyte.
Output >        e.g. 64k means 65536 bytes.
Output >
> Ending with exit status 0
```

JVM Path : /pkgs/lang/java/j2sdk1.4.2/jre/bin/java

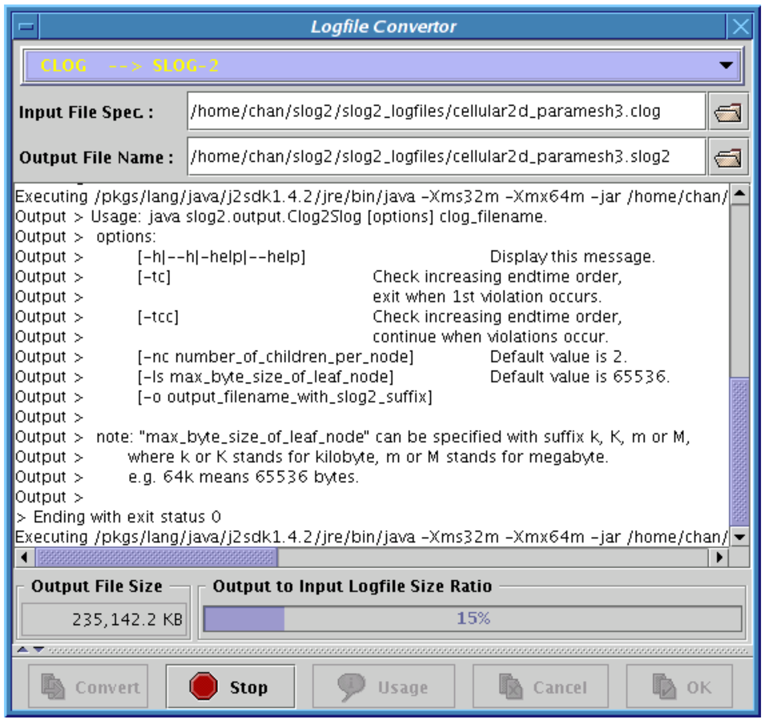
JVM Option : -Xms32m -Xmx64m

JAR Directory : /home/chan/slog2/build/lib

JAR Option :

TraceLibrary Path :

Figure 3.3: Hidden parameters panel of the Logfile Convertor.



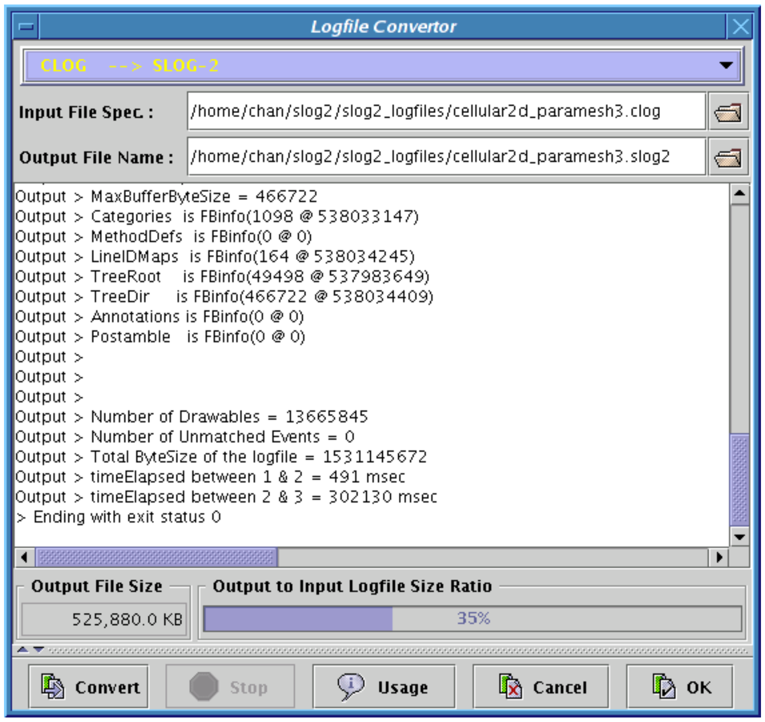
Input File Spec. : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.clog

Output File Name : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.slog2

```
Executing /pkgs/lang/java/j2sdk1.4.2/jre/bin/java -Xms32m -Xmx64m -jar /home/chan/
Output > Usage: java slog2.output.Clog2Slog [options] clog_filename.
Output >  options:
Output >         [-h|--h|-help|--help]                     Display this message.
Output >         [-tc]                            Check increasing endtime order,
Output >                                          exit when 1st violation occurs.
Output >         [-tcc]                           Check increasing endtime order,
Output >                                          continue when violations occur.
Output >         [-nc number_of_children_per_node]         Default value is 2.
Output >         [-ls max_byte_size_of_leaf_node]          Default value is 65536.
Output >         [-o output_filename_with_slog2_suffix]
Output >
Output >  note: "max_byte_size_of_leaf_node" can be specified with suffix k, K, m or M,
Output >        where k or K stands for kilobyte, m or M stands for megabyte.
Output >        e.g. 64k means 65536 bytes.
Output >
> Ending with exit status 0
Executing /pkgs/lang/java/j2sdk1.4.2/jre/bin/java -Xms32m -Xmx64m -jar /home/chan/
```

Output File Size: 235,142.2 KB

Figure 3.4: Logfile conversion in progress.



Input File Spec. : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.clog

Output File Name : /home/chan/slog2/slog2_logfiles/cellular2d_paramesh3.slog2

```
Output > MaxBufferByteSize = 466722
Output > Categories  is FBinfo(1098 @ 538033147)
Output > MethodDefs  is FBinfo(0 @ 0)
Output > LineIDMaps  is FBinfo(164 @ 538034245)
Output > TreeRoot    is FBinfo(49498 @ 537983649)
Output > TreeDir     is FBinfo(466722 @ 538034409)
Output > Annotations is FBinfo(0 @ 0)
Output > Postamble   is FBinfo(0 @ 0)
Output >
Output >
Output >
Output > Number of Drawables = 13665845
Output > Number of Unmatched Events = 0
Output > Total ByteSize of the logfile = 1531145672
Output > timeElapsed between 1 & 2 = 491 msec
Output > timeElapsed between 2 & 3 = 302130 msec
> Ending with exit status 0
```

Output File Size: 525,880.0 KB

Output to Input Logfile Size Ratio: 35%

Figure 3.5: The *OK* button is enabled when the logfile conversion finishes normally, i.e with exit status 0.

3.3 Legend Window

As soon as a SLOG-2 file is selected in the Main window and is ready for visualization, the Legend window like the one shown in Figure 3.6 will be displayed. All the features that are going to be discussed in the Legend window affect both the Timeline and the Histogram windows.

The Legend window contains mainly a seven-column legend table. The seven columns are labeled *Topo, Name, V, S, count, incl and excl* 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 3.7)	None
Name ▼	Name	Edit Name	String Sort Order Menu (Figure 3.8(a))
V ▼	Visibility	Check or Uncheck	Checkbox Operations Menu (Figure 3.9)
S ▼	Searchability	Check or Uncheck	Checkbox Operations Menu (Figure 3.9)
count ▼	Count	Non-editable	Number Sort Order Menu(Figure3.8(b))
incl ▼	Inclusion Ratio	Non-editable	Number Sort Order Menu(Figure3.8(b))
excl ▼	Exclusion Ratio	Non-editable	Number Sort Order Menu(Figure3.8(b))

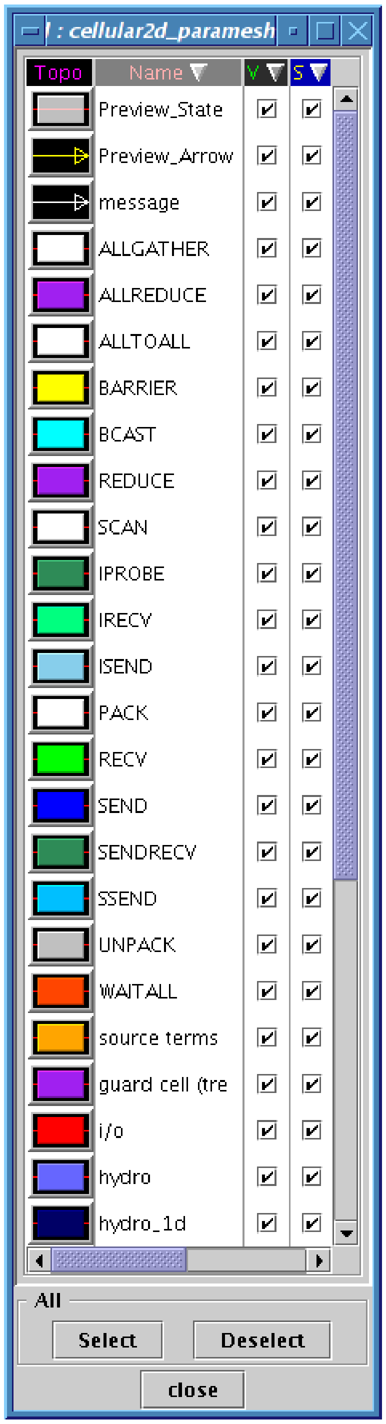
Table 3.6: Operations on the Legend window's columns.

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 and (2) right mouse clicking in any column cell.

Figure 3.7 is the Color Chooser dialog that will pop up when one of the icon buttons in column Topo is pressed. The color editor provides three 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; an explicit screen redraw is needed.

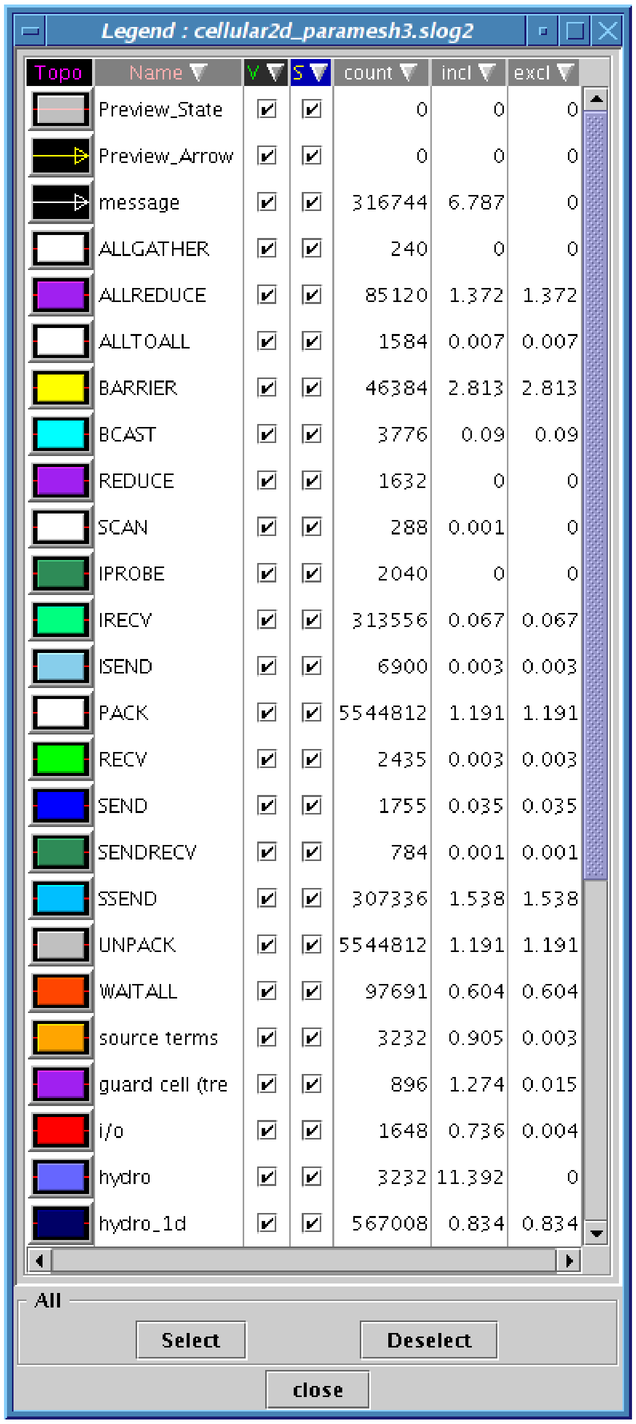
Figure 3.8(a) shows the popup dialog box either when the title icon of column *Name* is pressed or when the right mouse button is clicked somewhere in the column. Altogether, there are six different alphabetical sort orders as shown in the figure and they are summarized in Table 3.8. Fig 3.8(b) shows the popup dialog box for the column *count*, *incl* or *excl* is pressed or when the right mouse button is clicked somewhere in the column. The popup dialog box contains a menu which will be described later in Table 3.10.

The first four are various combinations of alphabetical and case-sensitive order; for example, *z...a Z...A* refers to a reverse-case-sensitive alphabetical ordering. The second-to-last order in the list is called the *Creation Order*, which refers to the order in which categories are stored in the slog2 file when they are being created. The four alphabetical orderings have two hidden sort orders. One is



Topo	Name	V	S
▭	Preview_State	☑	☑
→	Preview_Arrow	☑	☑
→	message	☑	☑
⬜	ALLGATHER	☑	☑
🟪	ALLREDUCE	☑	☑
⬜	ALLTOALL	☑	☑
🟨	BARRIER	☑	☑
🟦	BCAST	☑	☑
🟪	REDUCE	☑	☑
⬜	SCAN	☑	☑
🟩	IPROBE	☑	☑
🟩	IRECV	☑	☑
🟦	ISEND	☑	☑
⬜	PACK	☑	☑
🟩	RECV	☑	☑
🟦	SEND	☑	☑
🟩	SENDRECV	☑	☑
🟦	SSEND	☑	☑
▭	UNPACK	☑	☑
🟧	WAITALL	☑	☑
🟧	source terms	☑	☑
🟪	guard cell (tre	☑	☑
🟥	i/o	☑	☑
🟦	hydro	☑	☑
⬛	hydro_1d	☑	☑

(a) Legend Window's initial view



Topo	Name	V	S	count	incl	excl
▭	Preview_State	☑	☑	0	0	0
→	Preview_Arrow	☑	☑	0	0	0
→	message	☑	☑	316744	6.787	0
⬜	ALLGATHER	☑	☑	240	0	0
🟪	ALLREDUCE	☑	☑	85120	1.372	1.372
⬜	ALLTOALL	☑	☑	1584	0.007	0.007
🟨	BARRIER	☑	☑	46384	2.813	2.813
🟦	BCAST	☑	☑	3776	0.09	0.09
🟪	REDUCE	☑	☑	1632	0	0
⬜	SCAN	☑	☑	288	0.001	0
🟩	IPROBE	☑	☑	2040	0	0
🟩	IRECV	☑	☑	313556	0.067	0.067
🟦	ISEND	☑	☑	6900	0.003	0.003
⬜	PACK	☑	☑	5544812	1.191	1.191
🟩	RECV	☑	☑	2435	0.003	0.003
🟦	SEND	☑	☑	1755	0.035	0.035
🟩	SENDRECV	☑	☑	784	0.001	0.001
🟦	SSEND	☑	☑	307336	1.538	1.538
▭	UNPACK	☑	☑	5544812	1.191	1.191
🟧	WAITALL	☑	☑	97691	0.604	0.604
🟧	source terms	☑	☑	3232	0.905	0.003
🟪	guard cell (tre	☑	☑	896	1.274	0.015
🟥	i/o	☑	☑	1648	0.736	0.004
🟦	hydro	☑	☑	3232	11.392	0
⬛	hydro_1d	☑	☑	567008	0.834	0.834

(b) Fully extended Legend window that shows statistics sumary

Figure 3.6: Typical Legend window when a slog2 file is first loaded into Jumpshot-4. The Legend window can be expanded to reveal the hidden statistics, *count*, *inclusion ratio* and *exclusion ratio* of each category of drawables, stored in the 5th, 6th and 7th columns of the window.

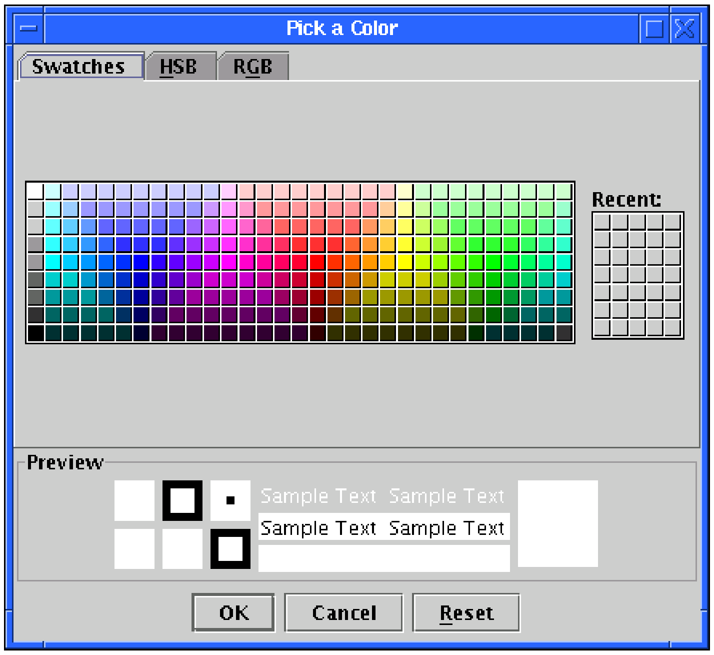
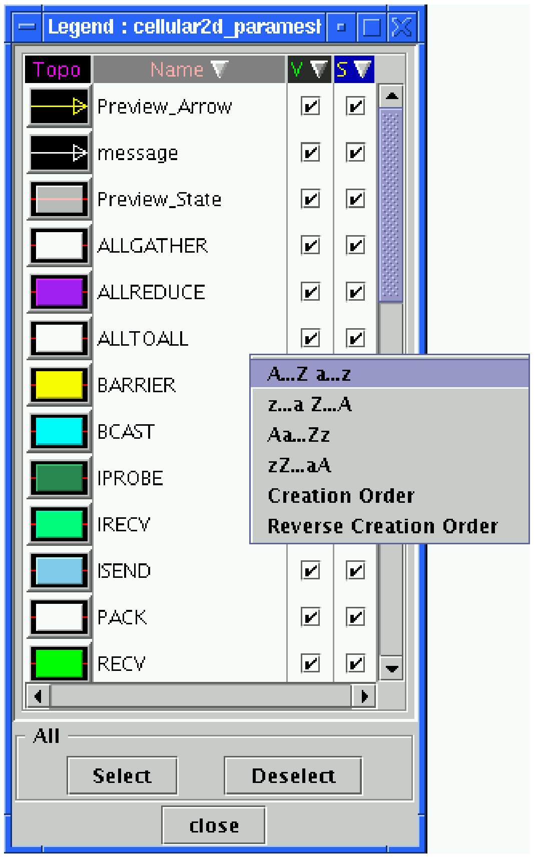
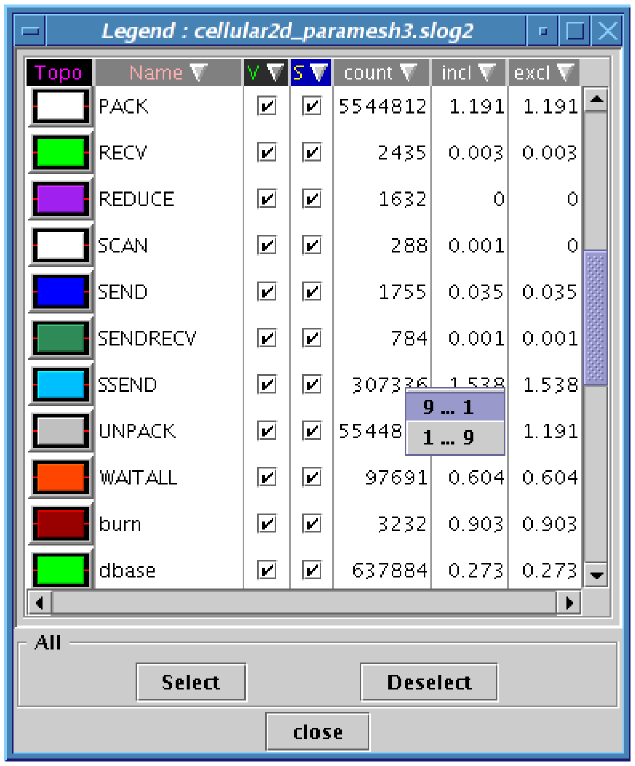


Figure 3.7: Color Chooser Dialog for column Category Topology



Topo	Name	V	S
▶	Preview_Arrow	✓	✓
▶	message	✓	✓
▨	Preview_State	✓	✓
⬜	ALLGATHER	✓	✓
🟪	ALLREDUCE	✓	✓
⬜	ALLTOALL	✓	✓
🟨	BARRIER		
🟦	BCAST		
🟩	IPROBE		
🟩	IRECV		
🟦	ISEND	✓	✓
⬜	PACK	✓	✓
🟩	RECV	✓	✓

(a) Sort Order menu for Name



Topo	Name	V	S	count	incl	excl
⬜	PACK	✓	✓	5544812	1.191	1.191
🟩	RECV	✓	✓	2435	0.003	0.003
🟪	REDUCE	✓	✓	1632	0	0
⬜	SCAN	✓	✓	288	0.001	0
🟦	SEND	✓	✓	1755	0.035	0.035
🟩	SENDRECV	✓	✓	784	0.001	0.001
🟦	SSEND	✓	✓	307336	1.538	1.538
▨	UNPACK	✓	✓	55448		1.191
🟧	WAITALL	✓	✓	97691	0.604	0.604
🟥	burn	✓	✓	3232	0.903	0.903
🟩	dbase	✓	✓	637884	0.273	0.273

(b) Sort Order menu for count, inclusion and exclusion ratios

Figure 3.8: Sort Order operation menu for the column *Name*, *count*, *incl* and *excl* 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: Description of the alphabetical Sort Order operation menu for column *Name* in the Legend window.

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 in Table 3.24.

Ordering	Description
9 ... 1	Decreasing numerical ordering
1 ... 9	Increasing numerical ordering

Table 3.10: Description of the numerical Sort Order operation menu for columns *count*, i*ncl* and *excl* in the Legend window

Table 3.10 shows the only two orderings allowed for all numerical entries in the Legend window.

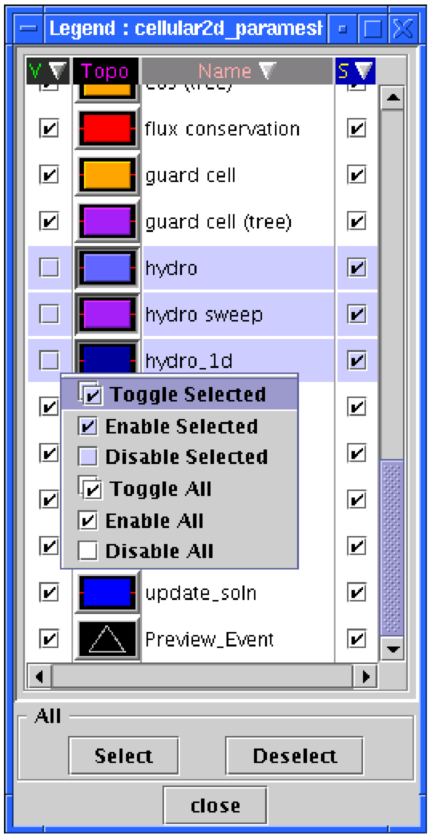


Figure 3.9: Checkbox Operation menu for column Category Visibility and Searchability

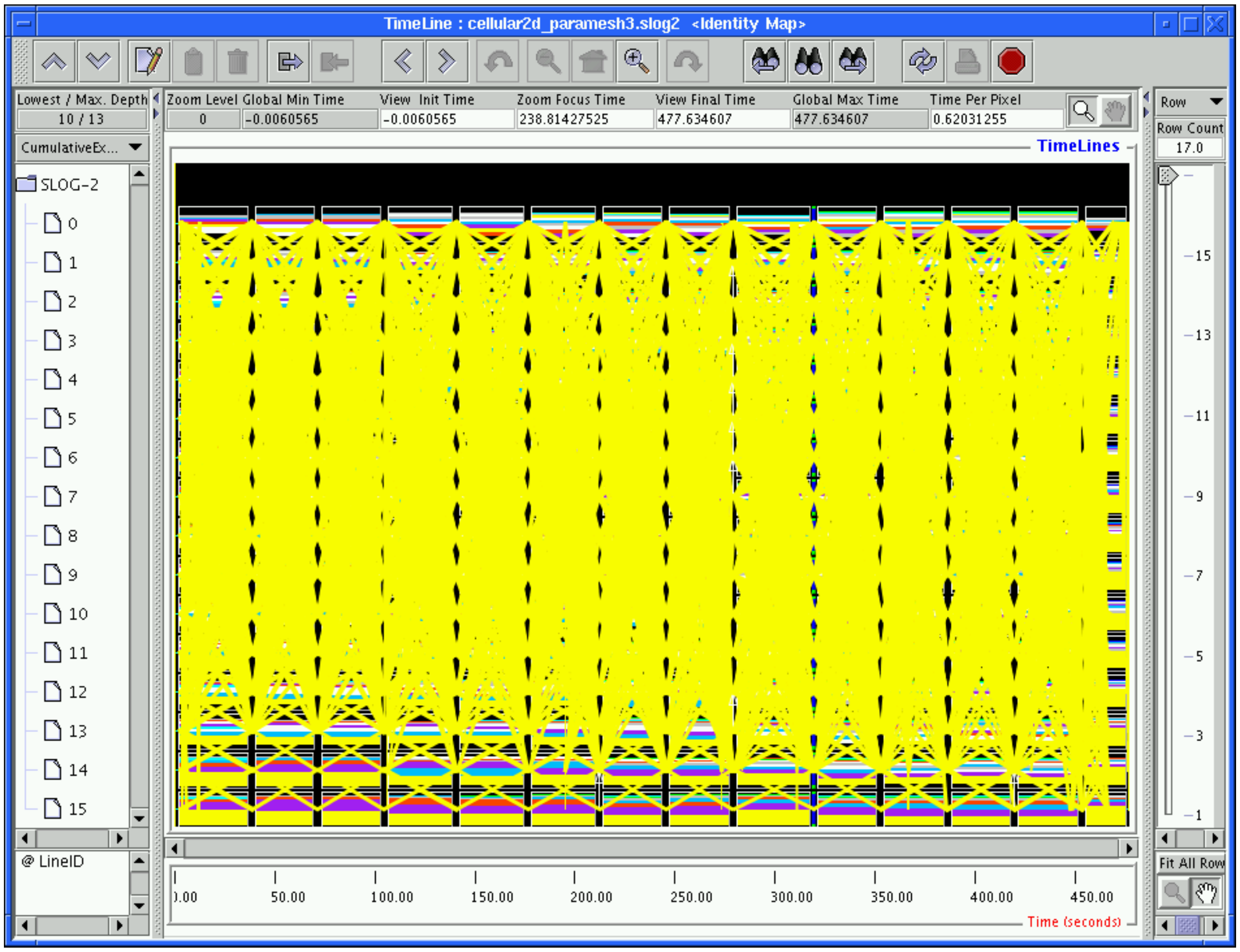
Figure 3.9 shows a popup dialog box when the title icon of column *V (Visibility)* or *S (Searchability)* is pressed or when the right mouse button is clicked somewhere in either column. The rule of case-sensitive alphabetical ordering allows all MPI names to be put before all user-defined categories. With continuous mouse selection, the user can easily toggle the visibility of user-defined states in the Timeline or Histogram window. Also, every element in the column Name can be edited. This feature allows the user to correct undesirable category names set during logfile creation and even facilitates 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 3.12: Standard selection rules.

NOTE: Any change done in the Legend window that alters the appearance of drawables will not be automatically updated in the Timeline canvas until the CanvasReDraw button in the Timeline window is pressed.

3.4 Timeline *Zoomable* Window



TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Lowest / Max. Depth: 10 / 13
Zoom Level: 0
Global Min Time: -0.0060565
View Init Time: -0.0060565
Zoom Focus Time: 238.81427525
View Final Time: 477.634607
Global Max Time: 477.634607
Time Per Pixel: 0.62031255
Row Count: 17.0

Figure 3.10: Initial display of the Timeline window of a 514 MB 16-process slog2 file with default preview resolution.

Most of the advanced features in the SLOG-2 viewer are provided through a *zoomable* window. Jumpshot-4 has two zoomable windows: Timeline and Histogram. Figure 3.10 is the initial display of the Timeline window of a half-gigabyte 16-timeline slog2 file. The zoomable window consists of several concealable and removable components. In the center of the window is the *zoomable and scrollable canvas*. For the Timeline window, the center canvas is called the *timeline canvas*. Directly on top of the zoomable canvas is the *time display panel*. On top of the display panel 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 the 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 three sides of the window. A bare-minimal zoomable window can be obtained by removing the toolbar and hiding the left and right panels. An almost-bare-minimal Timeline window looks like the one shown in Figure 2.1.

3.4.1 Zoomable and Scrollable Canvas

When a big slog2 file like the one shown in Figure 3.10is viewed, the whole timeline canvas is filled with preview drawables. Although it provides a reasonable description at a high level,[4] it is hard to know the details. Hence, a well-designed zoomable and scrollable user interface (ZSUI) of the timeline canvas becomes necessary to help the viewer locate events of interest. The ZSUI of the timeline canvas includes many parts and operations. 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 two such canvases in the Timeline window: *timeline canvas* and *time ruler canvas*. In these canvases, left mouse clicking can be alternated in two different modes by a pair of toggled buttons as shown in Figures 3.11 and 3.12. 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 at the end of the time display panel. The time ruler's toggled buttons are located at the bottom of row adjustment panel, 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 the 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 the scrollbar's knob, clicking the end buttons and in the space between the knob and scrollbar's end buttons.

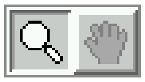


Figure 3.11: Canvas's left mouse click is in zoom mode.

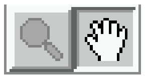


Figure 3.12: Canvas's left mouse click is in hand mode.

3.4.1.1 Dragged Zoom

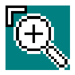


Figure 3.13: Zoom-plus cursor that indicates the left mouse clicking is ready for zooming in.

Dragged zoom is active only when the left mouse click is in zoom mode, that is, when the magnifying glass button is pressed in the toggled buttons as in Figure 3.11. In zoom mode, the cursor within the canvas will appear like a magnifying glass with a plus sign in the center, as in Figure 3.13. It is called the zoom-plus cursor. The dragged zoom operation is initialized by pressing the left mouse button at the beginning of the zoomed-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 zoomed-in region. The region that is marked by the pair of white lines is lightly shaded, as shown in Figure 2.4. The process can be canceled by hitting the ESC key during dragging. Once the left mouse button is released,

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

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 zoomed-in region. Notice that 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 3.14: 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; that is, the region centered at the point of left clicking will be magnified by a factor of 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 remainsa zoom-plus cursor. *Shift-click*, on the other hand, will do the opposite. While the shift key is held down, the cursor will be changed to a zoom-minus cursor as in Figure 3.14, 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

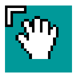


Figure 3.15: Open-hand cursor indicates that the left mouse click is ready to grasp and scroll.

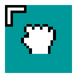


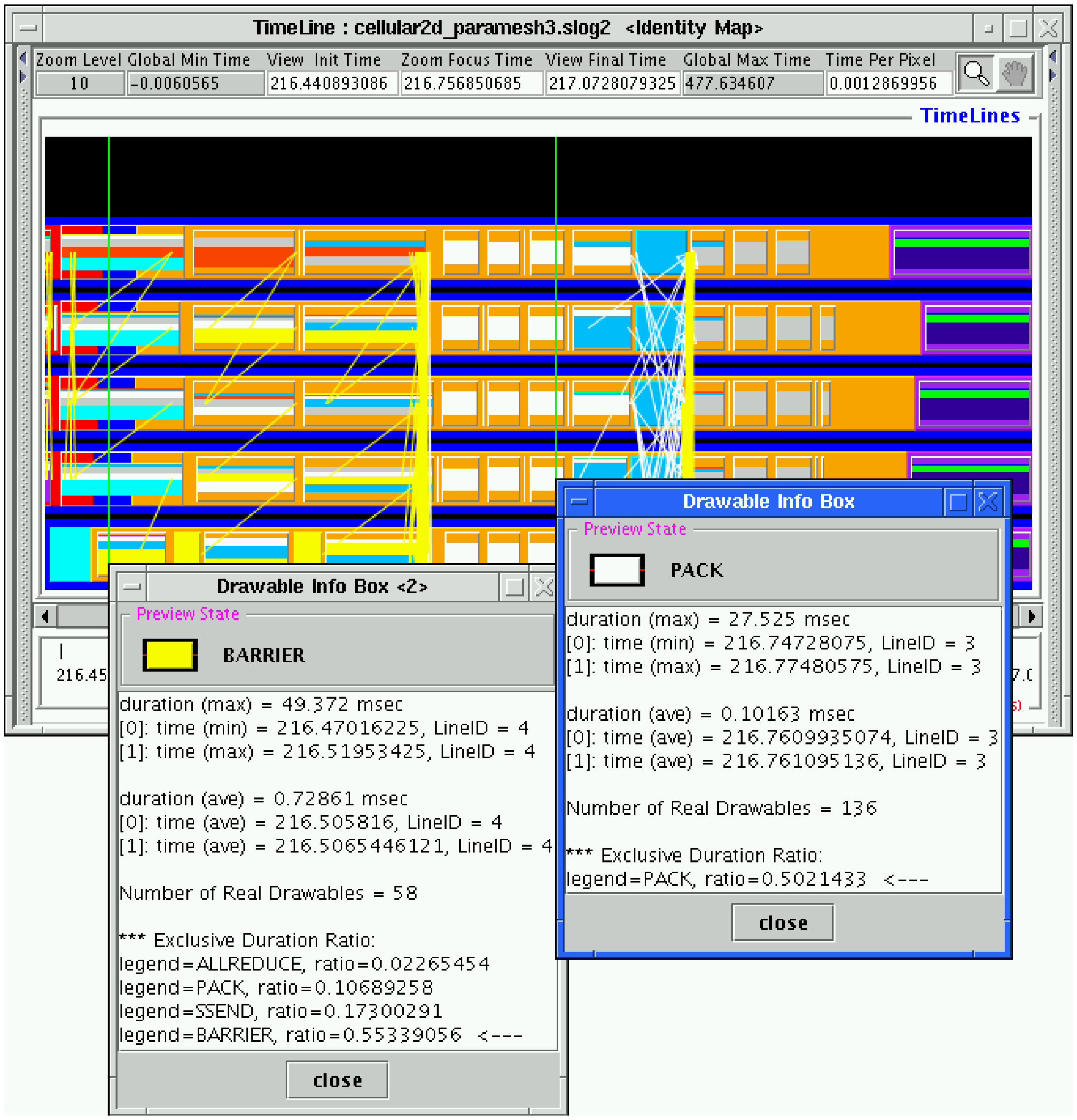
Figure 3.16: Closed-hand cursor indicates that the left mouse click is scrolling.

Grasp and Scroll is active only when the left mouse click is in hand mode, that is, when the open-hand button is pressed as in Figure 3.12. The cursor in hand mode is an open hand as in Figure 3.15. As soon as left mouse button is pressed down, the cursor turns to a closed hand, as in Figure 3.16. It indicates the canvas will move in the same direction that the cursor moves as long as the left mouse button remains pressed. The grasp and scroll mode in the time ruler canvas can move only horizontally, but the grasp and scroll mode in the timeline canvas allows movement both vertically and horizontally.

3.4.1.4 Information Dialog Box

To be complete, Jumpshot-4 provides a way to tell user exactly what is being displayed. This feature is particularly important when there are many preview drawables. Following standard user interface practice, Jumpshot-4 uses *right mouse clicking* as an interface for the user to tell Jumpshot-4 for what object more information is needed. In general, the user can inquire anywhere on the canvas (timeline or time ruler canvases) by right mouse clicking. An information dialog box will pop up accordingly to tell the user more about the object being clicked on. There are three 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 past or zoomed into. One of the uses of the info boxes is to serve as time markers in between zooming and scrolling.

Drawable Info Box The Drawable Info Box is a popup dialog box that provides detailed information about the drawable object that is being clicked on. There are two different kinds of Drawable Info Box: one for preview drawable and one for real drawable.



Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
10	-0.0060565	216.440893086	216.756850685	217.0728079325	477.634607	0.0012869956

```
Preview State
BARRIER
duration (max) = 49.372 msec
[0]: time (min) = 216.47016225, LineID = 4
[1]: time (max) = 216.51953425, LineID = 4

duration (ave) = 0.72861 msec
[0]: time (ave) = 216.505816, LineID = 4
[1]: time (ave) = 216.5065446121, LineID = 4

Number of Real Drawables = 58

*** Exclusive Duration Ratio:
legend=ALLREDUCE, ratio=0.02265454
legend=PACK, ratio=0.10689258
legend=SSEND, ratio=0.17300291
legend=BARRIER, ratio=0.55339056  <---
```

```
Preview State
PACK
duration (max) = 27.525 msec
[0]: time (min) = 216.74728075, LineID = 3
[1]: time (max) = 216.77480575, LineID = 3

duration (ave) = 0.10163 msec
[0]: time (ave) = 216.7609935074, LineID = 3
[1]: time (ave) = 216.761095136, LineID = 3

Number of Real Drawables = 136

*** Exclusive Duration Ratio:
legend=PACK, ratio=0.5021433  <---
```

Figure 3.17: Drawable Info Box for Preview State

Drawable Info Box for Preview Drawable Right mouse clicking on two of the preview states in the timeline canvas shown in Figure 2.9 will pop up two Drawable Info Boxes for the preview states. They are displayed in Figure 3.17. 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. To illustrate what information is presented by the Drawable Info Box, we take the highlighted Drawable Info Box in Figure 3.17 as an example. The Drawable Info Box for the 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 is a big text area that prints all the detailed statistical information about this preview state. There are six 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 three "average" timestamps are averaged over all the real drawables represented by this preview drawable. Besides timestamps, the info box also tells the"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 one category of real states in the preview state, so the 136 states are all PACKs. The sum of the durations of all PACKs is about half of the duration of the preview state, as indicated by "ratio=0.5021433".

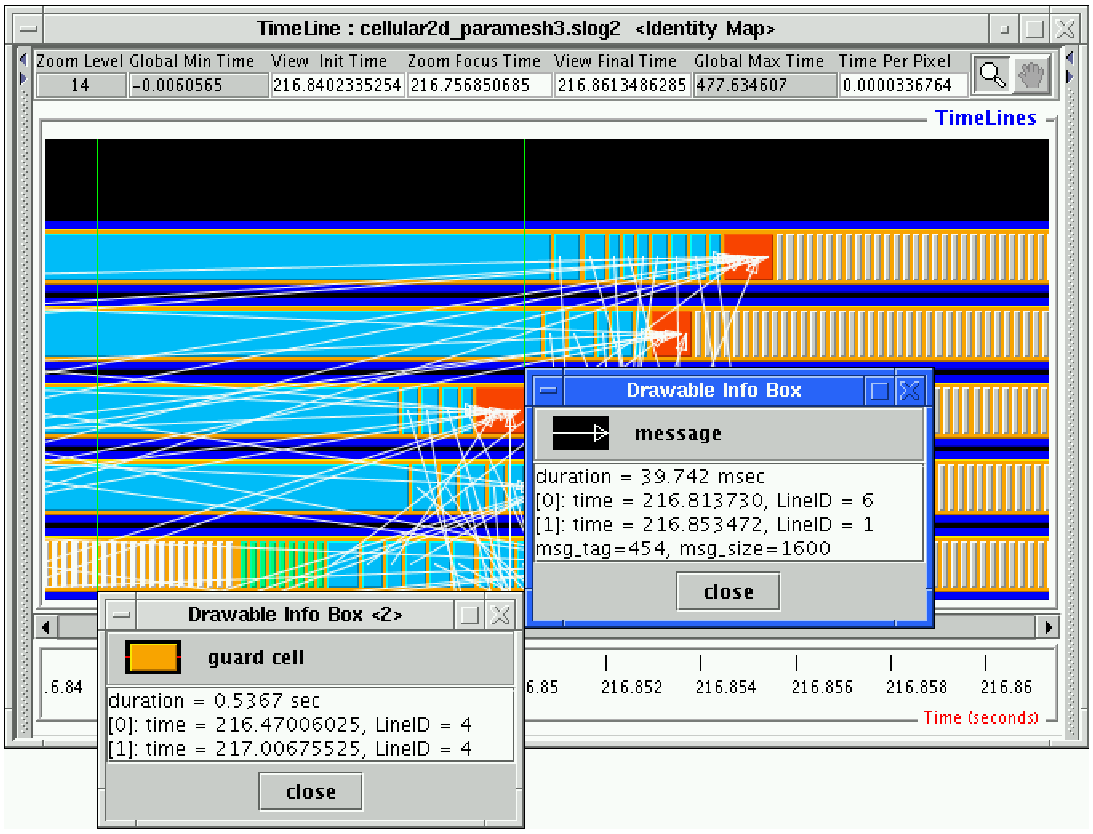
Another Drawable Info Box, shown in Figure 3.17, has its upper left hand pointed at a preview state that has four different strips of colors: yellow, royal blue, white, and purple. Right mouse clicking at the yellow strip pops up a Drawable Info Box with a yellow state icon with label BARRIER. As shown in the figure, this preview state amalgamated four categories of real states: ALLREDUCE, PACK, SSEND, and BARRIER; the statistically most significant one is BARRIER. It proportionally and exclusively occupies 55% of the length of the preview state. Hence the BARRIER strip is the tallest of all the color strips shown in the preview state. Clicking on a different color strip in the same preview state will pop up a drawable info box with a differently labeled icon, but the contents of the text area remains the same. In general, not every category listed in the text area is visible in the preview state display. Of the four categories mentioned in the text area, only three 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 the preview state in the Preference window that does not rely on the category ratio[5]. As indicated, there are 58 real drawables in the preview state, but no information is provided about how many real drawables are in each real category.

Drawable Info Box for Real Drawable Similarly for real drawables, the Drawable Info Box can be brought up by right mouse clicking on the real drawables. In Figure 3.18, 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.[6] The info box shows the starttime, start timeline ID, endtime, and ending timeline ID, as well as some extra information implemented by the native format. In this example, the extra information is the message size carried by the specific arrow. The message size is 1600 bytes.

Duration Info Box The 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

[5]That is, by setting the PREVIEW_STATE_DISPLAY pulldown menu in the Timeline window or Preference window to the *FitMostLegends* message, as listed in Table 3.20 .

[6]The vicinity width can be adjusted by modifying the parameter CLICK_RADIUS_TO_LINE in the Preference window as listed in Table 3.18 The default is 3 pixels.



TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
14	-0.0060565	216.8402335254	216.756850685	216.8613486285	477.634607	0.0000336764

Drawable Info Box

message

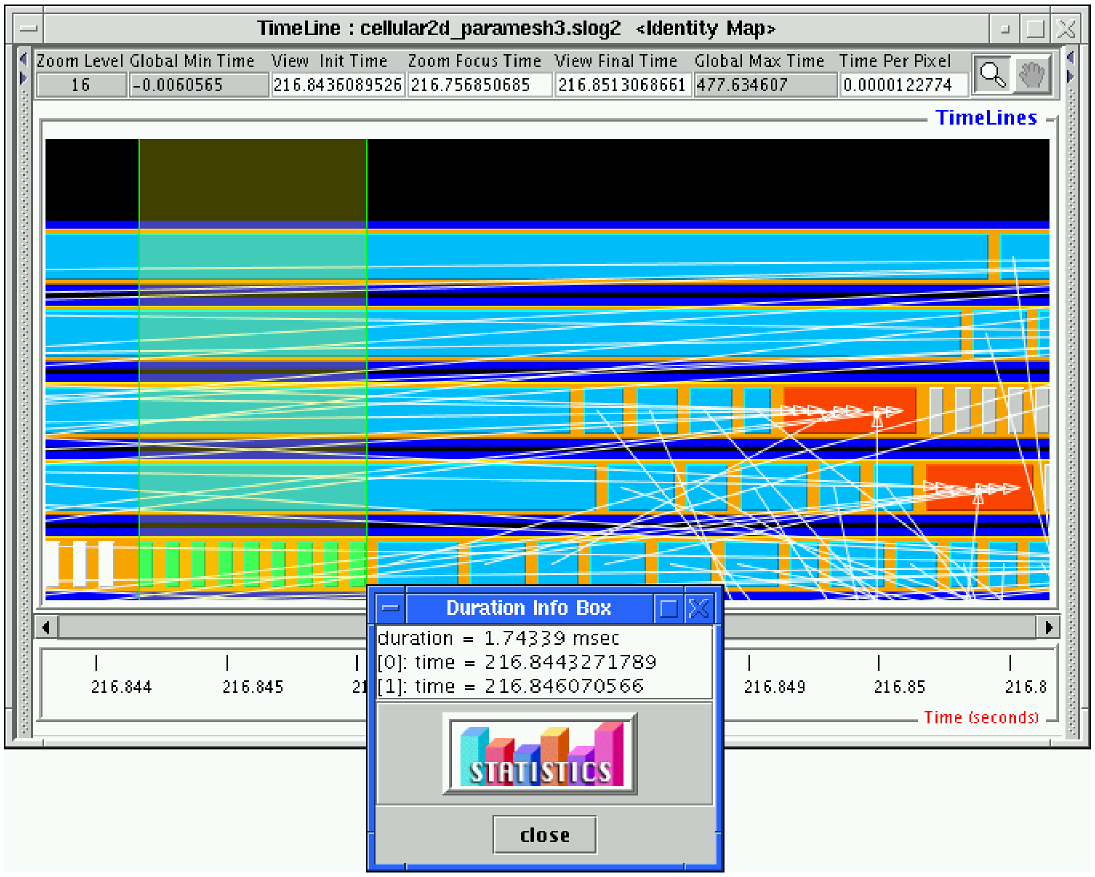
```
duration = 39.742 msec
[0]: time = 216.813730, LineID = 6
[1]: time = 216.853472, LineID = 1
msg_tag=454, msg_size=1600
```

Drawable Info Box <2>

guard cell

```
duration = 0.5367 sec
[0]: time = 216.47006025, LineID = 4
[1]: time = 217.00675525, LineID = 4
```

Figure 3.18: Drawable Info Box for real state and arrow. The Drawable Info Box for the arrow shows the message size, 1600 bytes, and tag ID, 454.

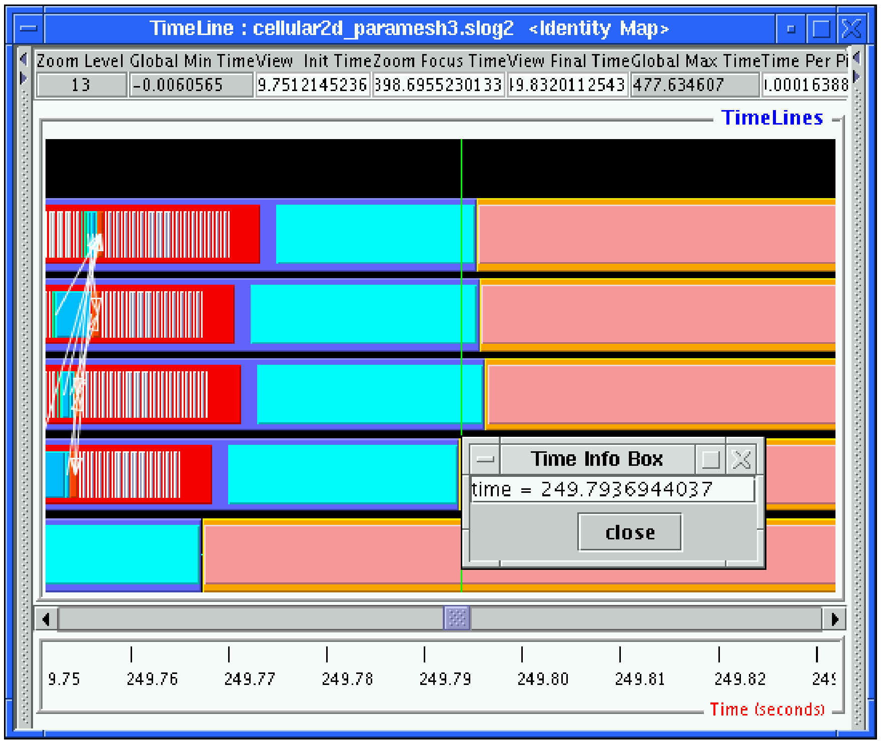


Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
16	-0.0060565	216.8436089526	216.756850685	216.8513068661	477.634607	0.0000122774

```
duration = 1.74339 msec
[0]: time = 216.8443271789
[1]: time = 216.846070566
```

Figure 3.19: Duration Info Box shows the duration, starttime, and endtime of a time region marked by a pair of green lines.

lines and is lightly shaded as well. The Duration Info Box can serve a marker to facilitate the process of zooming in and out. The information provided by Duration Info Box can also be used to compare different durations or to measure the total duration of a collection of subroutine calls. For instance, in Figure 3.19, the Duration Info Box marks all consecutive green states on the fifth timelines. The Duration Info Box says the total duration of the nine green states is about 1.74 msec.



TimeLine : cellular2d_paramesh3.slog2 <Identity Map>

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pi
13	-0.0060565	9.7512145236	398.6955230133	19.8320112543	477.634607	1.00016388

Time Info Box

time = 249.7936944037

Figure 3.20: Time Info Box displays the time of where it pops up.

Time Info Box Time Info Box is created by right clicking in the empty space in either the timeline or the time ruler canvas, as in Figure 3.20. This Info Box is usually used as a marker for a single event in time.

3.4.2 Toolbar

The buttons in the toolbar of the Timeline window provide various basic services to the Timeline window. Table 3.14 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 the y-axis label panel. It contains a tree-like representation for the y-axis label for the timelines. For a SLOG-2 file convertible from CLOG, CLOG-2 or RLOG with the default viewmap, the typical y-axis label panel looks like that shown in Figure 3.21. Together with the toolbar's label buttons (e.g., LabelMark and LabelMove) and standard mouse selection methods listed in Table 3.12, labels can be rearranged easily to create a more easily understood timeline canvas. For the multiple-viewmaps SLOG-2 file from IBM's UTE

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. Useful in Process-Thread or Communicator-Thread view to expand each process timeline into multiple thread timelines in multi-threaded slog2 file.
⇦	LabelCollapse	Alt-C	Collapse the y-axis tree label by 1 level. Useful in collapsing multiple thread timelines into single process timeline.
<	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 3.14: Toolbar functionalities.

trace environment, the 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; that is, the user needs to press the CanvasReDraw button in the toolbar to update the Timeline window with the changes from the label panel.

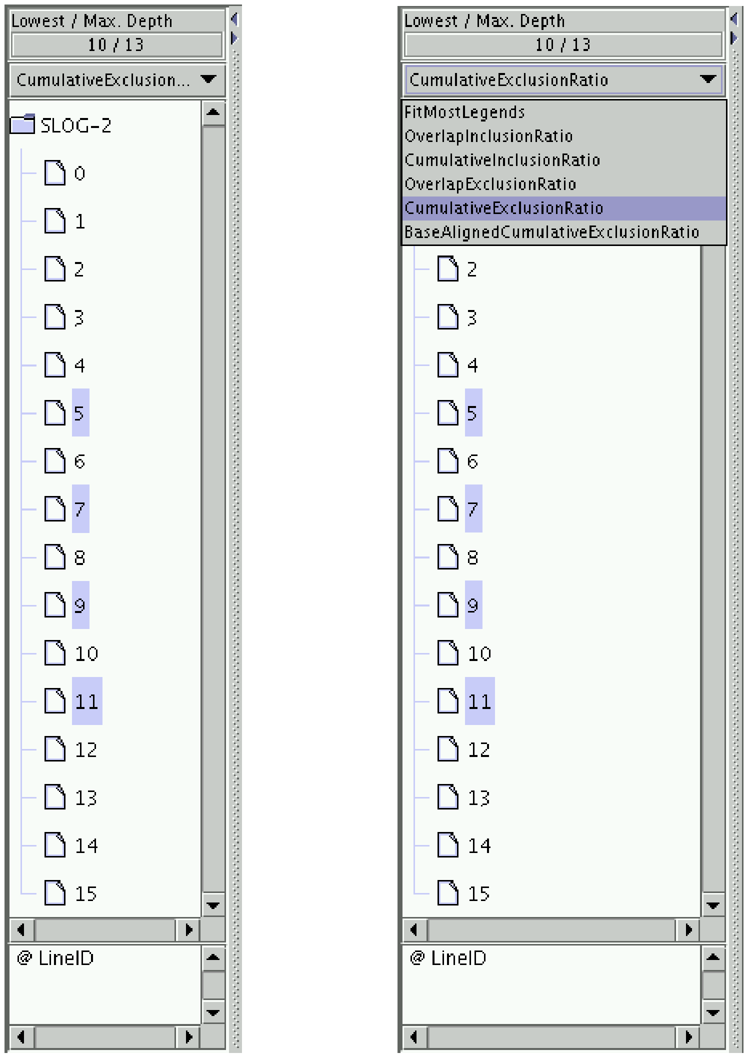
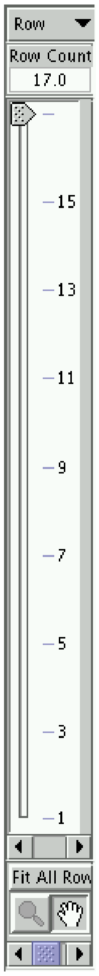


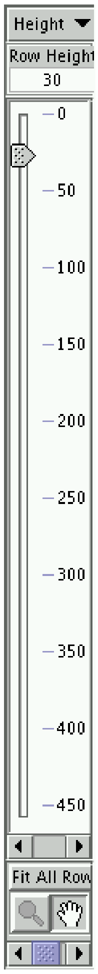
Figure 3.21: Simple one-level y-axis label tree. The blue highlighted labels are those that have been selected. The pulldown menu at the top of panel indicates the value in PREVIEW_STATE_DISPLAY in the Preference window.

3.4.4 Row Adjustment Panel

The concealable right panel in the Timeline window contains the row adjustment panel, which is used to determine the row adjustment scheme. There are two different modes in the row adjustment panel: row count mode and row height mode. These two 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 the number 1.[7] 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[8]. For a 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 the user hits the CanvasReDraw button. Together with window resize, the row adjustment panel allows the user to magnify or shrink the height of the timeline as desired.

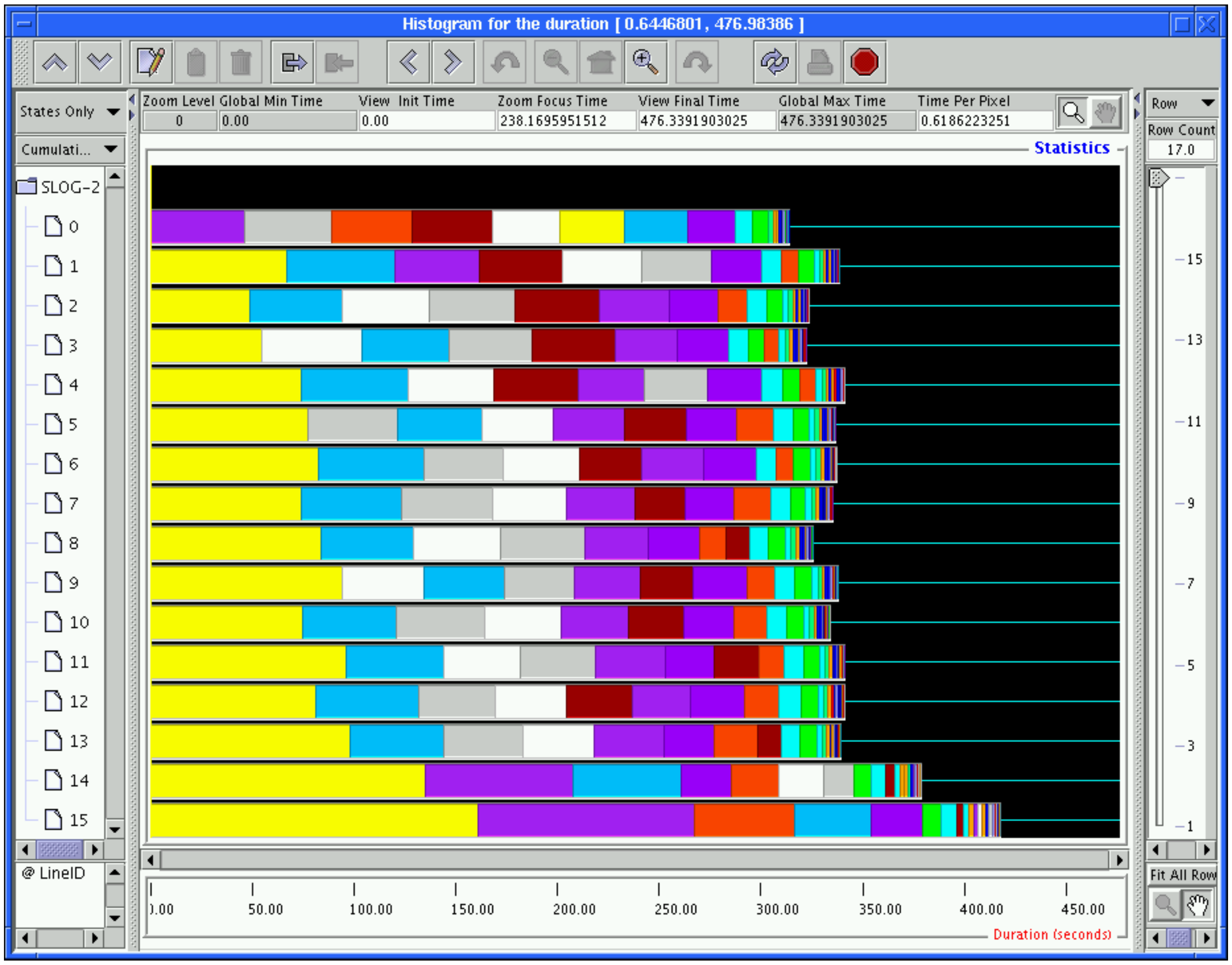


(a) Row Count mode



(b) Row Height mode

Figure 3.22: Row Adjustment Panel determines the Timeline window's resize scheme. When one of the mode sliders or text fields is adjusted, the other three components will be adjusted simultaneously.



Zoom Level 0
Global Min Time 0.00
View Init Time 0.00
Zoom Focus Time 238.1695951512
View Final Time 476.3391903025
Global Max Time 476.3391903025
Time Per Pixel 0.6186223251
Row Count 17.0

Figure 3.23: Histogram window of the whole duration shown in Figure 3.10.

3.5 Histogram *Zoomable* Window

The Histogram window is created by clicking the statistics button located in the middle of Duration Info Box, shown in Figure 3.19. In Figure 3.23, the Histogram window is created for the whole duration of the timeline canvas in Figure 3.10, that is, the same duration as the complete 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 3.23 that the yellow state (MPI_Barrier in this case) cumulatively takes up the most time. This is especially true in the last timeline.

Since the Histogram window is also a *zoomable* window like the Timeline window, a lot of the features described in Section 3.4.1 for the Timeline window are available for the Histogram window as well, for example, dragged-zoom, grasp and scroll, instant zoom in/out, easy vertical expansion of timeline, and cut and paste of timelines. If some state categories or timelines need to be made invisible in the Histogram window, one can disable the corresponding categories in the Legend window's column V or S or selected corresponding timelines in the Histogram window. The process is just like that for the Timeline window.

Only summary objects can be displayed in the Histogram window. Summary objects are similar to preview objects discussed earlier. However, whereas Preview objects are created during the logfile creation stage and cannot be modified during visualization, Summary objects are created dynamically during visualization, that is, during creation of a Duration Info Box, so they can be modified easily by the user. There are two different kinds of summary objects: summary state and summary arrow. There is only one summary state per timeline and one summary arrow for each ordered pair of timelines.[9] Currently three different views are available in the Histogram window: *States Only*, *Arrows Only,* and *All*. In the *States Only* view, only summary states are displayed. In the *Arrows Only* view, only summary arrows are displayed. In the *All* view, both summary states and arrows are displayed.

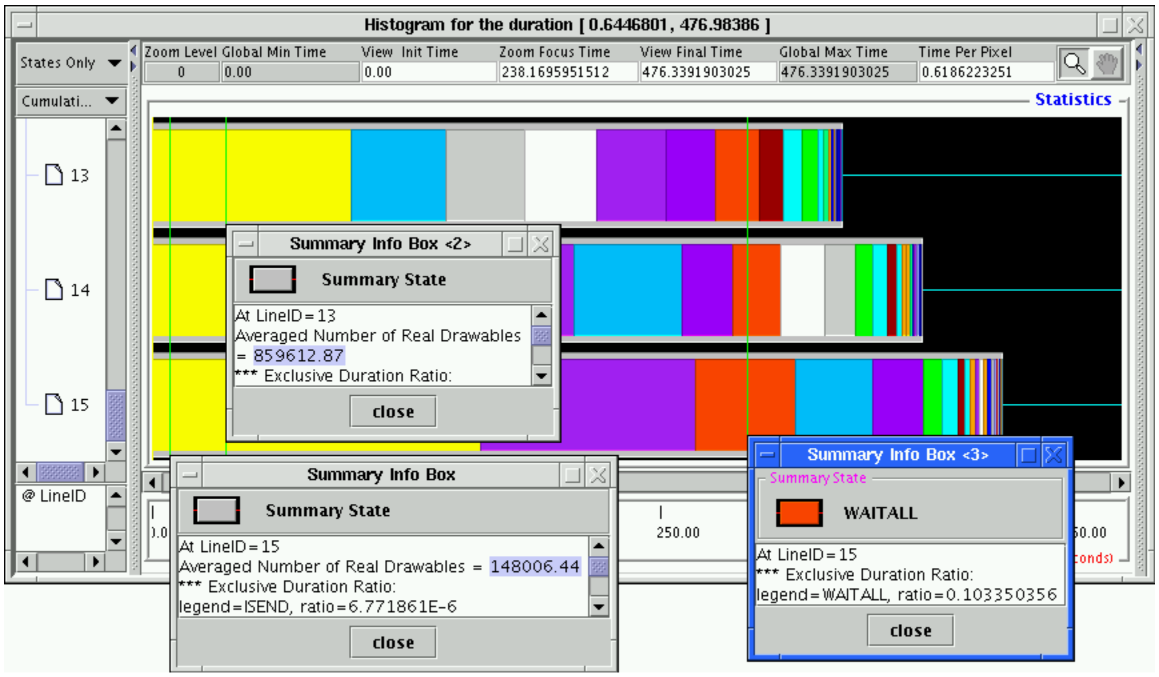
3.5.1 Summary States

Since summary states are created through the statistics of real and preview states, summary states the inherit properties of preview states, that is, inclusion and exclusion ratios. Hence, different representations of summary state are formed based on the PREVIEW_STATE_DISPLAY discussed in Section 2.2.1. Different representations of summary states can be selected through the SUMMARY_STATE_DISPLAY pulldown menu located at the top of the left panel in the histogram window or through a similar variable defined in the Preference window and in Table 3.22. Figure 3.23 is actually a *CumulativeExclusionRatio* view. Since the most time-consuming timeline is the last one, we will zoom in on the last three timelines and use them to discuss the visual representation of summary state. Figure 3.24 shows the last three timelines of Figure 3.23. Each summary state has a gray bordered box. Right mouse clicking at the bordered box pops up the Summary Info Box for

[7]If the slog2 file contains numerous timelines, increasing the row height will increase the size of the images managed by Jumpshot-4. This action 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 if there isn't enough physical memory in the machine that Jumpshot-4 runs on.

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

[9]An ordered pair of timelines means that the timeline pair (1,2) is different from the pair (2,1).



Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0	0.00	0.00	238.1695951512	476.3391903025	476.3391903025	0.6186223251

Summary Info Box <2>

Summary State

At LineID=13
Averaged Number of Real Drawables
= 859612.87
*** Exclusive Duration Ratio:

Summary Info Box

Summary State

At LineID=15
Averaged Number of Real Drawables = 148006.44
*** Exclusive Duration Ratio:
legend=ISEND, ratio=6.771861E-6

Summary Info Box <3>

Summary State

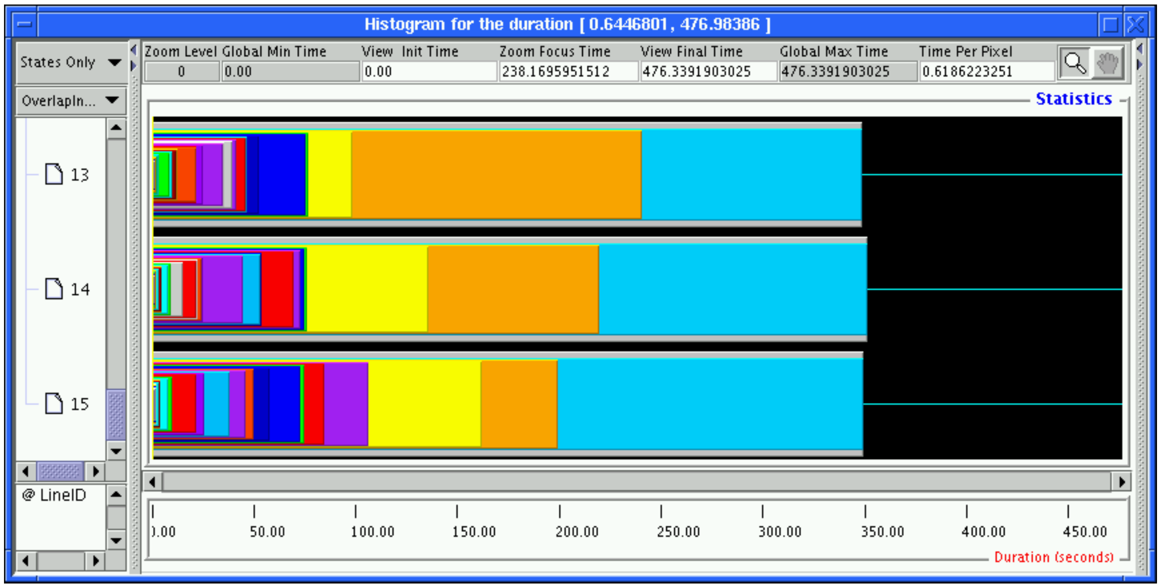
WAITALL

At LineID=15
*** Exclusive Duration Ratio:
legend=WAITALL, ratio=0.103350356

Figure 3.24: Summary state info boxes of the Histogram window.

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 148,006 real states and the timeline 13 summary state has about 859,613 real states; that is, timeline 13 has 5.8 times the number of real 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 on 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; see, for example, the highlighted summary info box in the Figure 3.24. The remaining duration at the end of each timeline is unaccounted for. In this particular logfile, the remaining time could be thought of as being used for computation.

Switching the SUMMARY_STATE_DISPLAY pulldown menu in the Histogram window in the figure to *OverlapInclusionRatio* redraws the histogram canvas. The histogram canvas now looks like the one shown in Figure 3.25. Since the sum of all inclusion ratios is greater than 1.0, the *CumulativeInclusionRatio* view is not provided in the Histogram window.[10] All the member categories of the summary states in the *OverlapInclusionRatio* view are drawn from the beginning of the histogram canvas and are nested one inside the others in decreasing inclusion ratio order, so the largest inclusion ratios are easily noticeable. To see the smallest ratios, one needs to zoom in around the beginning of the canvas. In Figure 3.25, the largest inclusion ratios in the three visible timelines are all royal blue and take up about the same amount of time. The second largest ratios are all orange colored and smallest in the timeline 15. Therefore, the *OverlapInclusionRatio* is good for comparing member category contribution among different timelines.



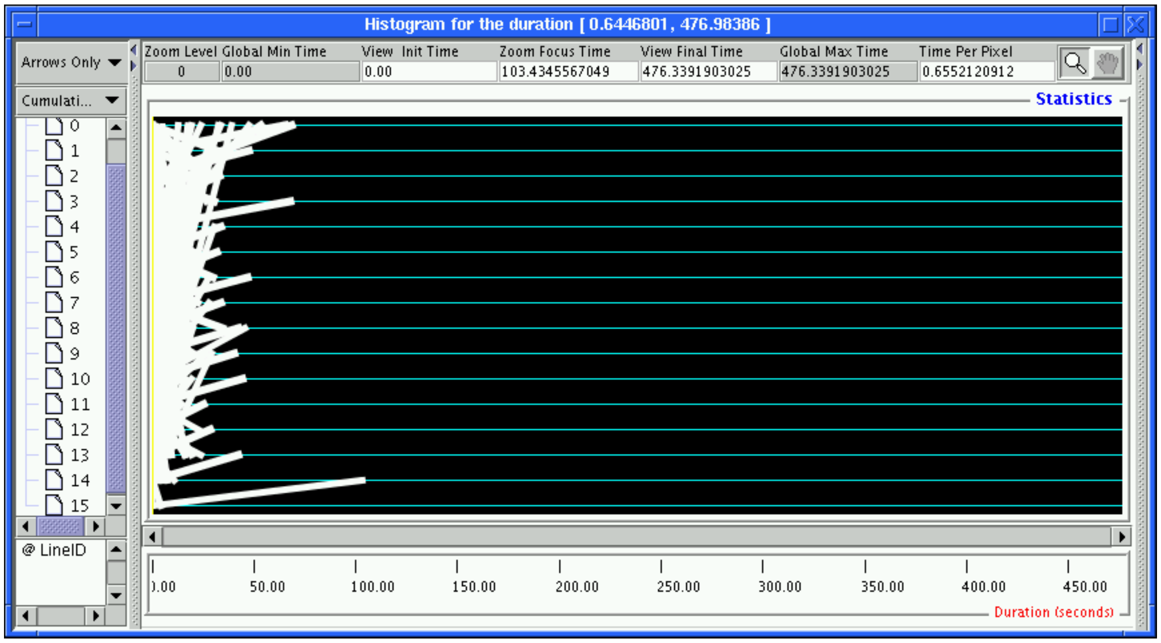
Histogram for the duration [0.6446801 , 476.98386]

States Only

OverlapIn...

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0	0.00	0.00	238.1695951512	476.3391903025	476.3391903025	0.6186223251

Figure 3.25: *OverlapInclusionRatio* view of Figure 3.24.



Histogram for the duration [0.6446801 , 476.98386]

Arrows Only

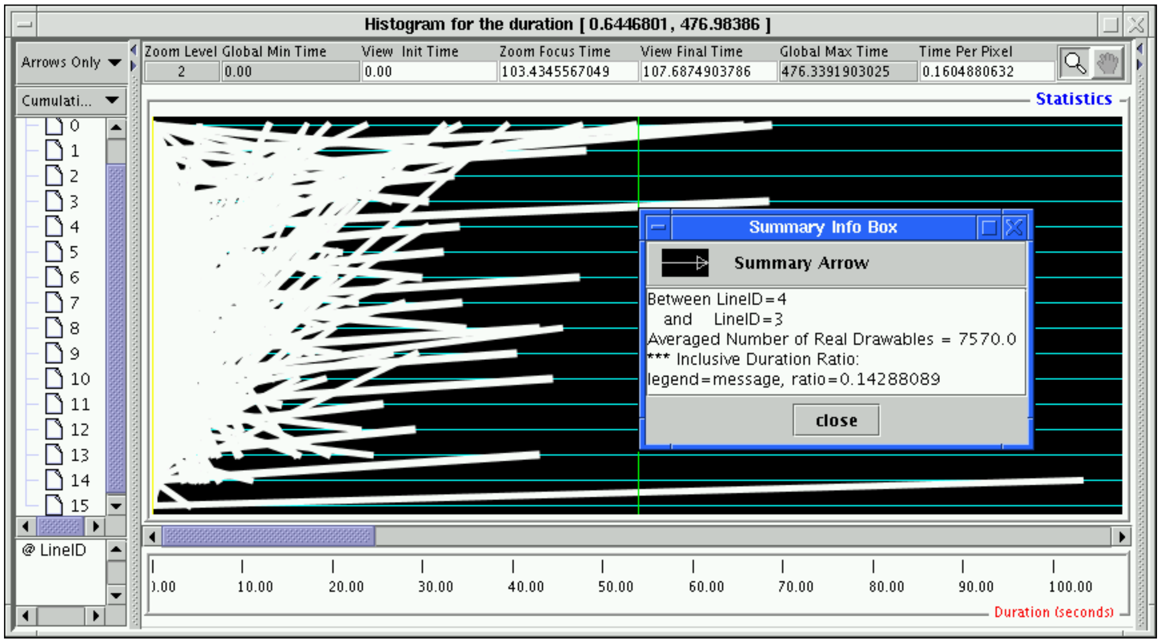
Cumulati...

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0	0.00	0.00	103.4345567049	476.3391903025	476.3391903025	0.6552120912

Figure 3.26: *Arrows Only* view of the Figure 3.23.

3.5.2 Summary Arrows

Figure 3.26 is the *Arrows Only* view of the histogram window shown in Figure 3.23. There is a summary arrow per ordered pair of timelines. The duration of each summary arrow is the total duration of all real arrows taking place between the ordered pair of timelines within the duration of the canvas. Notice that the duration of summary arrow may be longer than that of the canvas.


Histogram for the duration [0.6446801 , 476.98386]

Zoom Level: 2
Global Min Time: 0.00
View Init Time: 0.00
Zoom Focus Time: 103.4345567049
View Final Time: 107.6874903786
Global Max Time: 476.3391903025
Time Per Pixel: 0.1604880632

Summary Info Box

Summary Arrow

Between LineID=4
  and   LineID=3
Averaged Number of Real Drawables = 7570.0
*** Inclusive Duration Ratio:
legend=message, ratio=0.14288089

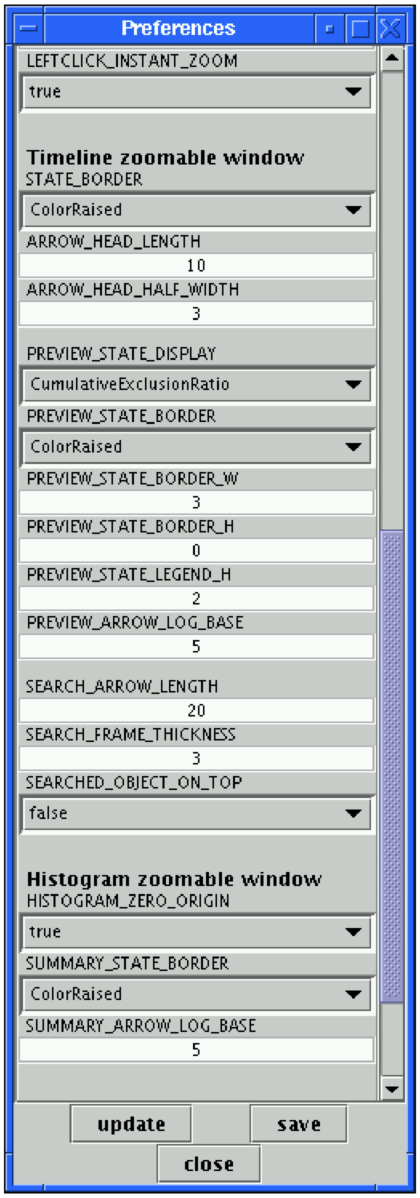
Figure 3.27: Arrow Summary Info Box of Figure 3.26.

Right mouse clicking at the summary arrow will display a Summary Info Box for the arrow as in the Figure 3.27. 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, the summary arrow provides a way to tell which ordered pair of timelines communicates the most.

3.6 Preference Window

As shown in Figure 3.28, the Preference window adjusts the various display properties of the visualization program. The parameters and their definitions are listed in Tables 3.16, 3.18, 3.20, 3.22 , and 3.24.

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



LEFTCLICK_INSTANT_ZOOM
true

Timeline zoomable window
STATE_BORDER
ColorRaised
ARROW_HEAD_LENGTH
10
ARROW_HEAD_HALF_WIDTH
3

PREVIEW_STATE_DISPLAY
CumulativeExclusionRatio
PREVIEW_STATE_BORDER
ColorRaised
PREVIEW_STATE_BORDER_W
3
PREVIEW_STATE_BORDER_H
0
PREVIEW_STATE_LEGEND_H
2
PREVIEW_ARROW_LOG_BASE
5

SEARCH_ARROW_LENGTH
20
SEARCH_FRAME_THICKNESS
3
SEARCHED_OBJECT_ON_TOP
false

Histogram zoomable window
HISTOGRAM_ZERO_ORIGIN
true
SUMMARY_STATE_BORDER
ColorRaised
SUMMARY_ARROW_LOG_BASE
5

Figure 3.28: Preference window showing the PREVIEW_STATE_DISPLAY.

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 an 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 3.16: Parameters for the section of *Zoomable Window Reinitialization* in the 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 pixels to be considered a dragged operation.
CLICK_RADIUS_TO_LINE	+ve integer	Radius in pixels 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 3.18: Parameters for the section of *All Zoomable Windows* in the 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 pixels.
ARROW_HEAD_HALF_WIDTH	+ve integer	Half-width of arrow head's base in pixels.
PREVIEW_STATE_DISPLAY	FitMostLegends, OverlapInclusionRatio, CumulativeInclusionRatio, OverlapExclusionRatio, CumulativeExclusionRatio, BaseAlignedCumulative-ExclusionRatio	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 pixels for the Preview state.
PREVIEW_STATE_BORDER_H	integer	The empty border insets' height in pixels for the Preview state.
PREVIEW_STATE_LEGEND_H	integer	Minimum height of the legend division (category strip) in pixels 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 pixels
SEARCH_FRAME_THICKNESS	integer	Thickness in pixels 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 3.20: Parameters for the section of *Timeline Zoomable Window* in the 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 3.22: Parameters for the section of *Histogram Zoomable Window* in the 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 3.24: Parameters for the section of the*Legend window* in the Preference window.

Chapter 4

Special Features

4.1 Search and Scan Facility

The level-of-detail support provided in SLOG-2 and Jumpshot-4's timeline window tends to help locate states that either are longer in time or occur very frequently. States that are short and occur rarely in a big logfile are difficult to locate without a special tool. User can easily spot the rarest states from looking at column *count* in the Legend window as in Figure 3.6. In Jumpshot-4, a search and scan facility is provided to facilitate this goal. There are three search criteria: search time, searchable timeline IDs, and searchable categories.

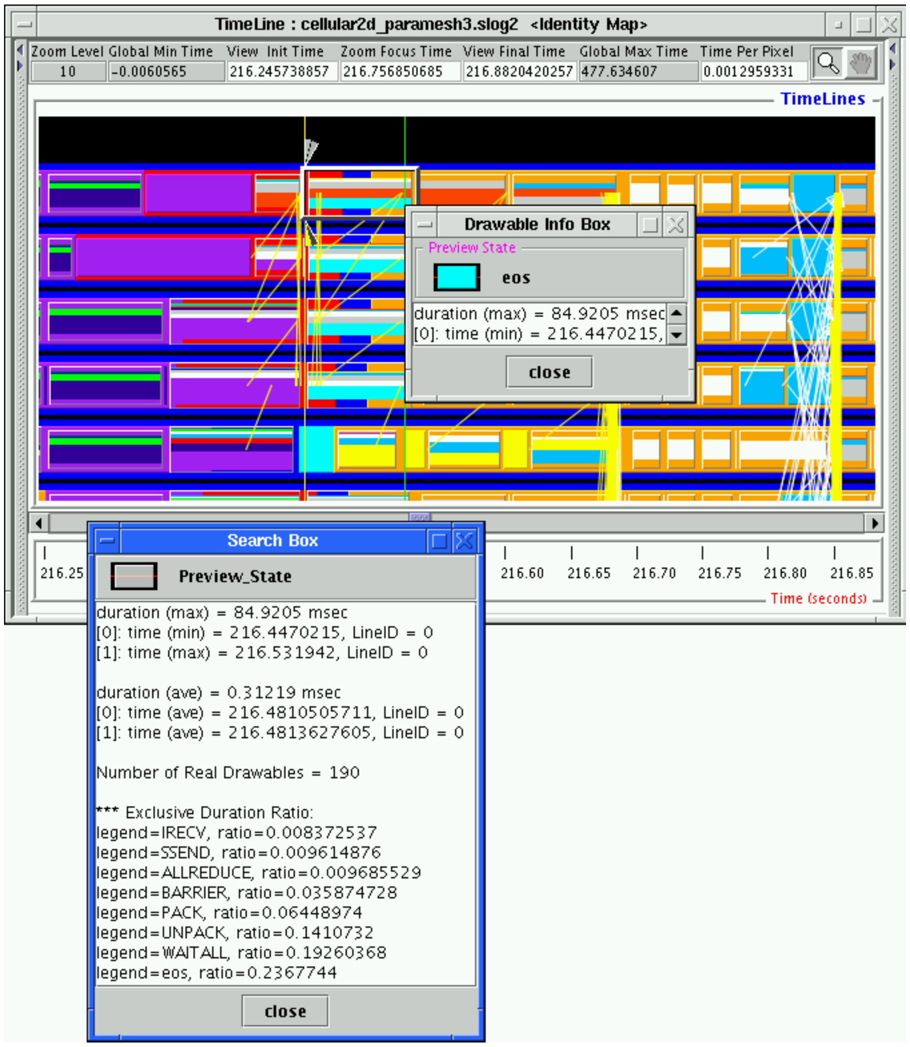
1. *Search time* is the time that search starts. It is marked by a yellow line called the search cursor. There are two different ways of setting the search cursor. When the timeline canvas is in hand mode, as described in Figure 3.12 of Section 3.4.1, left mouse clicking will set the search cursor. The other way can be done in either hand or zoom mode. First, one pops up an information dialog box of any kind, using right mouse clicking; then one presses the SearchInitialize button in the toolbar to replace the green line by the yellow search cursor. When more than one information dialog box is shown, the information dialog box shown last will have its green line used to initialize the search cursor. When the Timeline window first starts up, the search cursor is set at the starttime of the logfile.

2. *Searchable timeline IDs* are the timelines that the search will operate on; 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 3.12. When nothing is selected, all timelines are searchable.

3. *Searchable categories* are categories that have their searchable checkboxes enabled as in Figure 3.9. Only a drawable with a searchable category can be returned by the search facility. By default, all categories in the Legend window are searchable.

After any needed search criteria have been set, the search operation can be carried out by pressing either the SearchForeward or SearchBackward buttons shown in Table 3.14. As shown in Figure 4.1, the search facility returns a searched state that is marked by a transparent [1]box with a 3D

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

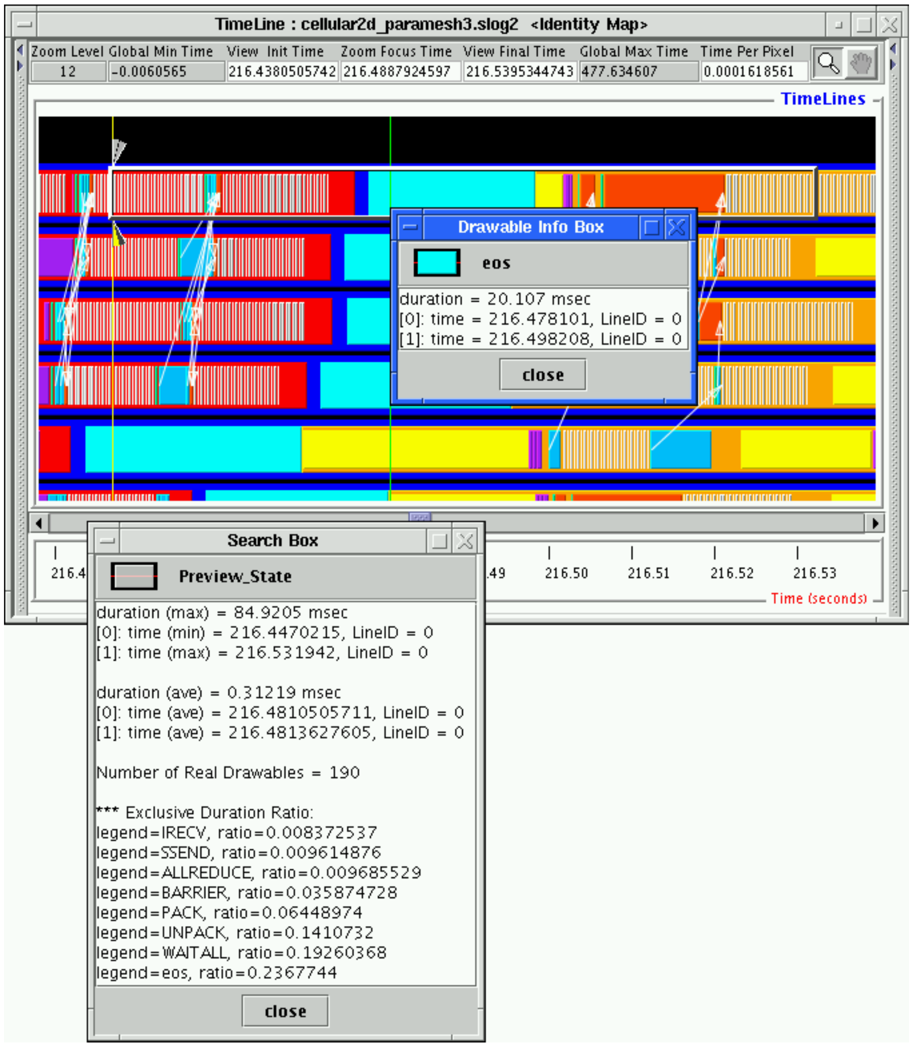


duration (max) = 84.9205 msec
[0]: time (min) = 216.4470215,

```
duration (max) = 84.9205 msec
[0]: time (min) = 216.4470215, LineID = 0
[1]: time (max) = 216.531942, LineID = 0

duration (ave) = 0.31219 msec
[0]: time (ave) = 216.4810505711, LineID = 0
[1]: time (ave) = 216.4813627605, LineID = 0

Number of Real Drawables = 190

*** Exclusive Duration Ratio:
legend=IRECV, ratio=0.008372537
legend=SSEND, ratio=0.009614876
legend=ALLREDUCE, ratio=0.009685529
legend=BARRIER, ratio=0.035874728
legend=PACK, ratio=0.06448974
legend=UNPACK, ratio=0.1410732
legend=WAITALL, ratio=0.19260368
legend=eos, ratio=0.2367744
```

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.

raised border and whose starttime is marked by a yellow search cursor and an upper and a lower 3D arrowhead. The upper 3D arrow's color matches that of the returned state. In the figure, the returned state is a preview state, so the upper 3D arrow is gray, as shown in the Legend window. Accompanied with the 3D raised bordered box is a popup Search Box that shows the details of the preview state, like the Drawable Info Box in Figure 3.17. Since the search in the figure is looking for state *eos*, a Drawable Info Box is shown to indicate that 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; 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 to by the Drawable Info Box.



Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
12	-0.0060565	216.4380505742	216.4887924597	216.5395344743	477.634607	0.0001618561

Drawable Info Box

eos

```
duration = 20.107 msec
[0]: time = 216.478101, LineID = 0
[1]: time = 216.498208, LineID = 0
```

close

Search Box

Preview_State

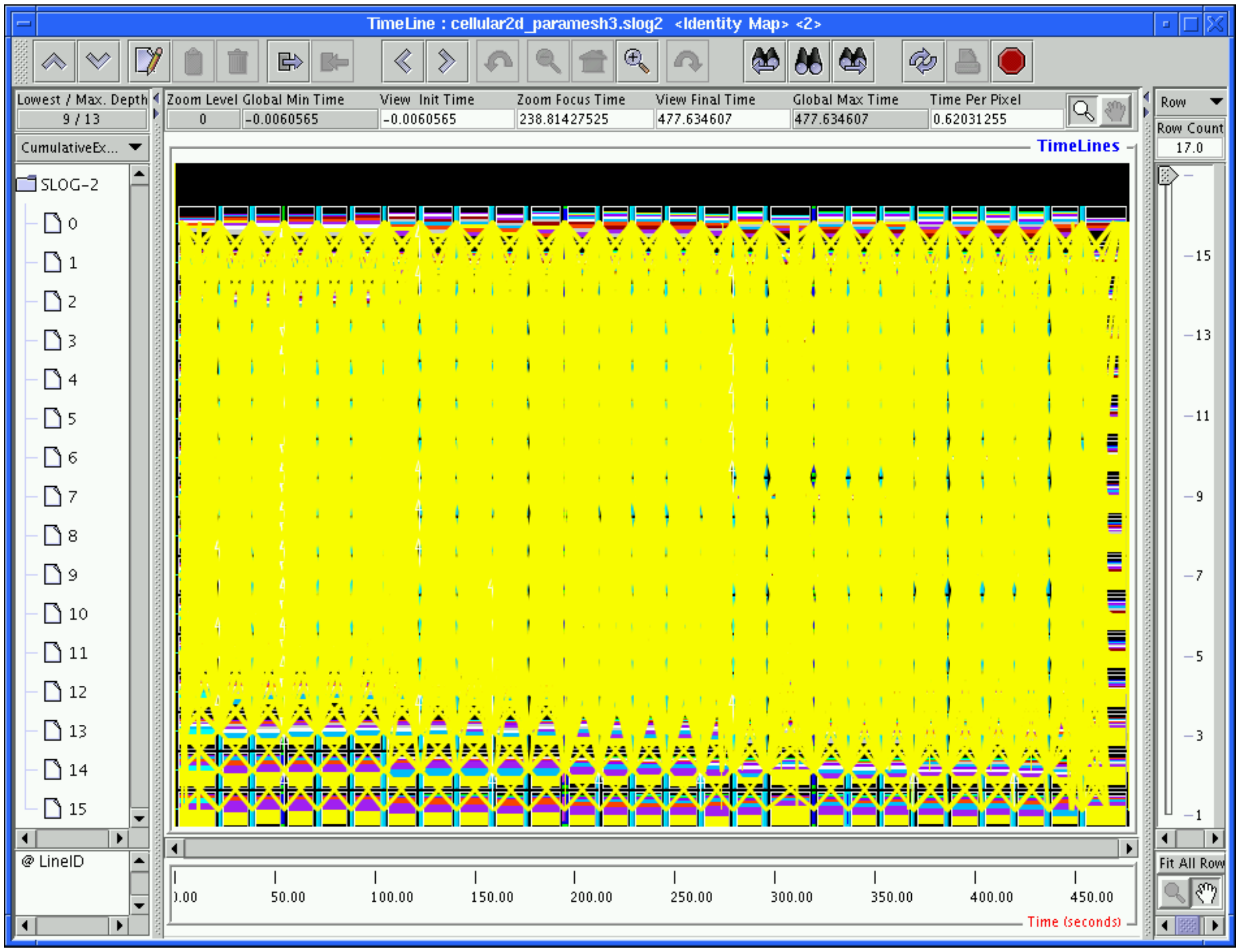
```
duration (max) = 84.9205 msec
[0]: time (min) = 216.4470215, LineID = 0
[1]: time (max) = 216.531942, LineID = 0

duration (ave) = 0.31219 msec
[0]: time (ave) = 216.4810505711, LineID = 0
[1]: time (ave) = 216.4813627605, LineID = 0

Number of Real Drawables = 190

*** Exclusive Duration Ratio:
legend=IRECV, ratio=0.008372537
legend=SSEND, ratio=0.009614876
legend=ALLREDUCE, ratio=0.009685529
legend=BARRIER, ratio=0.035874728
legend=PACK, ratio=0.06448974
legend=UNPACK, ratio=0.1410732
legend=WAITALL, ratio=0.19260368
legend=eos, ratio=0.2367744
```

close

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

In general, when one is searching in a big slog2 file, all preview categories should be set searchable; otherwise, searching for real drawables may not return anything because at the lower zoom level there may be no real drawables of the categories of interest. Only the preview drawable contains the categories of interest. Also, the search facility is carried out for the drawables that are in the physical memory. On rare occasions, drawables in the memory may have been exhausted for searching before the end of the logfile has been reached; thus, the user may need to advance the search by scrolling forward or backward to read in more drawables and to restart the search. 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 a later version of Jumpshot-4.

4.2 Tuning of the Timeline Window


TimeLine : cellular2d_paramesh3.slog2 <Identity Map> <2>

Lowest / Max. Depth: 9 / 13
Zoom Level: 0
Global Min Time: -0.0060565
View Init Time: -0.0060565
Zoom Focus Time: 238.81427525
View Final Time: 477.634607
Global Max Time: 477.634607
Time Per Pixel: 0.62031255
Row Count: 17.0

Figure 4.3: Initial Timeline window with finer preview resolution.

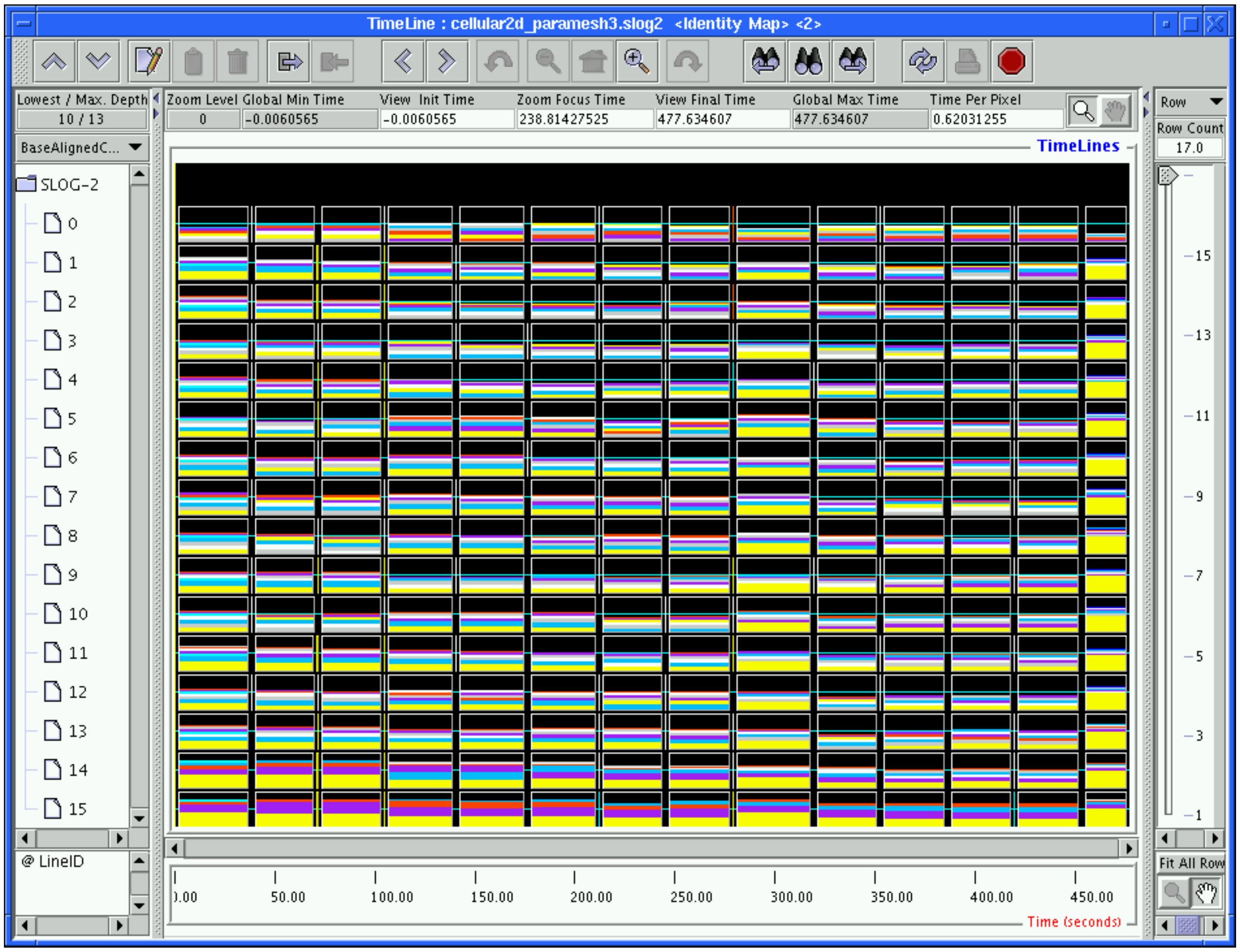
One of the major improvements in the new Jumpshot and SLOG-2 is the scalability in terms of visualization performance. As shown in Figure 3.10, 14 bands of preview states cover the whole canvas in the initial Timeline window. By incrementing the parameter INIT_SLOG2_LEVEL_READ by 1 in the Preference window as in Table 3.16, the initial Timeline window is redisplayed, as shown in Figure 4.3. The new Timeline window has 27 bands of preview states instead of 14. The increased preview resolution offers a more detailed description of the logfile but at the expense of graphical performance because every increment of INIT_SLOG2_LEVEL_READ roughly doubles the number of drawables to be iterated during every zooming or scrolling.[2] The biggest demand of graphical performance occurs when 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,

[2]It is true for a binary tree.

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 that significantly affects the graphical performance is ARROW_ANTIALIASING in the 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 a high-quality picture is needed, for example during screen capture for a picture or when anti-aliasing lines are drawn with graphics hardware support, turning on ARROW_ANTIALIASING is not recommended.

4.3 Estimation of MPI Communication Overhead



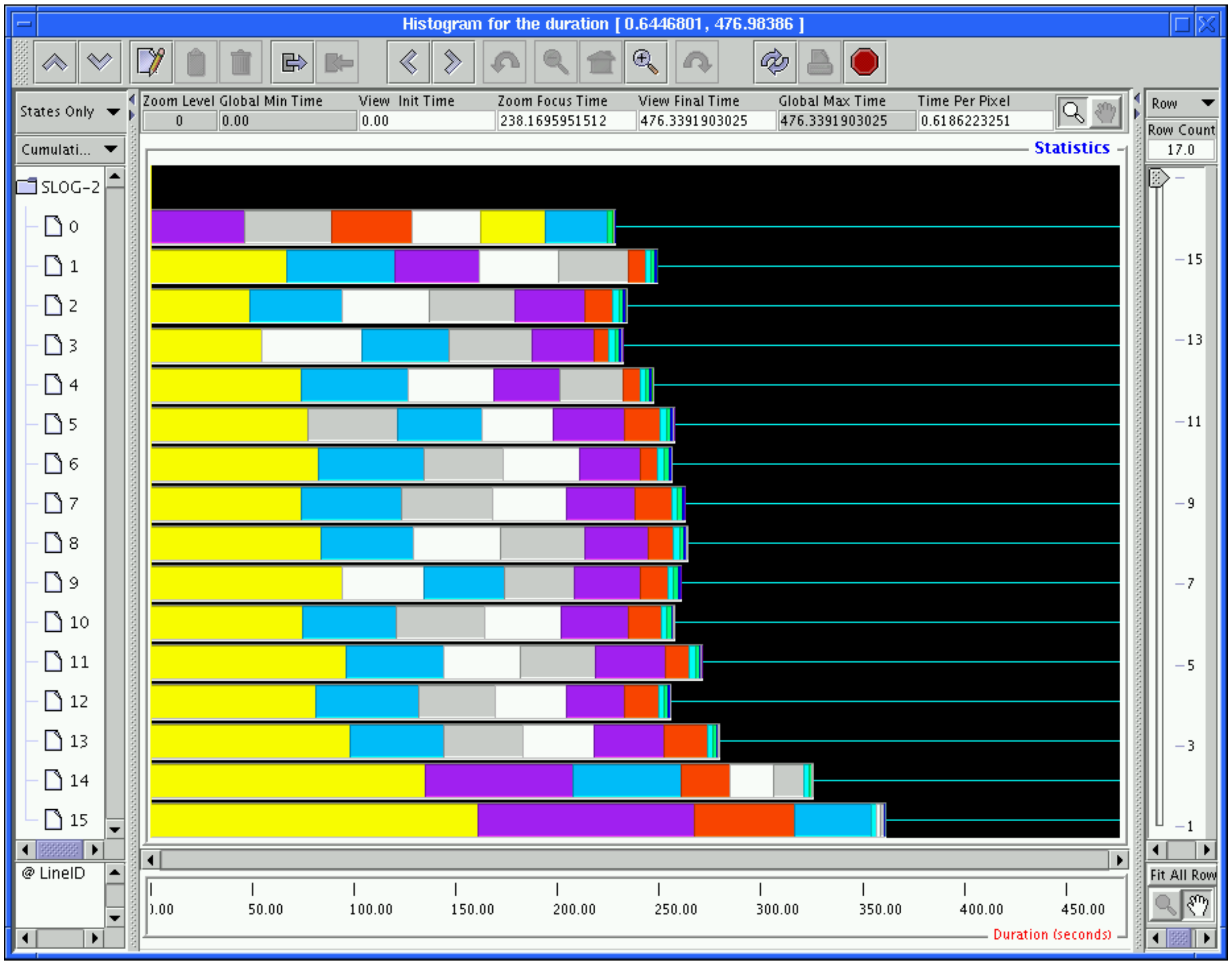
Lowest / Max. Depth: 10 / 13
Zoom Level: 0
Global Min Time: -0.0060565
View Init Time: -0.0060565
Zoom Focus Time: 238.81427525
View Final Time: 477.634607
Global Max Time: 477.634607
Time Per Pixel: 0.62031255
Row Count: 17.0

Figure 4.4: Graphical MPI overhead profiling through the use of the *BaseAlignedCumulativeExclusionRatio* view of Timeline window in Figure 3.10.

Most MPI application developers want to know about the overhead of MPI calls in their programs. Essentially, they want to know what the communication overheadis in their parallel programs. New SLOG-2 viewer provides a graphical answer to this question for most MPI profiling systems. In

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

MPE profiling systems, MPI states are alway nested deeper than the user-defined states. Therefore, disabling the user-defined states and arrows in the *CumulativeExclusionRatio* mode in the Timeline window still leaves all MPI exclusion ratios intact, without distorting the collective meaning of exclusion ratios. Figure 4.4 shows a *CumulativeExclusionRatio* view in *BaseAligned* mode that looks like a two-dimensional projection of a three-dimensional histogram for a timeline vs time coordinate system. The base aligned feature is for easy comparison of preview states' heights. From the figure, we know that the yellow state (i.e., MPI_Barrier) takes the most time in the program; we also know when and where MPI_Barrier consumes the most time. The combination of disabling user-defined states and using the *BaseAlignedCumulativeExclusionRatio* Timeline view provides a powerful and convenient way to estimate MPI communication overhead. Together with the zoomable capability of the Timeline window, the 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, as shown in Figure 4.5. The empty region in each timeline is assumed to be for user computation.


Histogram for the duration [0.6446801, 476.98386]

Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0	0.00	0.00	238.1695951512	476.3391903025	476.3391903025	0.6186223251

States Only
Cumulati...
Row
Row Count 17.0
Statistics
Duration (seconds)

Figure 4.5: Overall MPI overhead histogram for Figure 4.4.

4.4 Performance Analysis of Threaded MPI Application

The goal of this section is to provide an example how Jumpshot's ViewMap can be used to study the performance of threaded MPI applications. Let's say we are interested to find out the performance of different MPI implementations in a threaded environment. We will use a simple mult-threaded MPI program to see if there is any preformance difference. The test program, pthread_sendrecv.c, used here first creates multiple threads in each process. Each spawned thread then duplicates the MPI_COMM_WORLD to form a ring, i.e. each thread sends a message to its next rank and receives a message from its previous rank within same duplicated MPI_COMM_WORLD to form a ring. The program is shown at the end of the document. MPE is built with –enable-threadlogging[4] and –disable-safePMPI[5]. The most accessible MPI implementations with MPI_THREAD_MULTIPLE support are MPICH2 and OpenMPI. We will use the latest stable release of MPICH2, 1.0.5p4, and OpenMPI, 1.2.3 for this demonstration. Since OpenMPI has the option to enable progress thread in additional to the standard thread support, we will build 2 different versions of OpenMPIs for this little experiment. The experiment will be performed on 4 AMD64 nodes running Linux Ubuntu 7.04, each node consists of 4 cores and the test program will be running with 1 to 6 extra threads to see if the oversubscribing has any effect on the send and receive performance.

Table 4.2 shows the total duration of the 4-process run with various numbers of child threads. The data shows that as the number of child threads increases, so is the total runtime. For MPICH2, the runtime increase is modest for each additional thread. For OpenMPI+progress_thread, the performance isn't as good as MPICH2 but it is still reasonable as the number of threads increases. However for OpenMPI without progress thread support, the runtime increases drastically as there are 3 child threads or more. The situation becomes very bad as the node becomes oversubscribed, i.e. when there are 5 or more child threads. Now we are going to use MPE logging and Jumpshot to find out what happens.

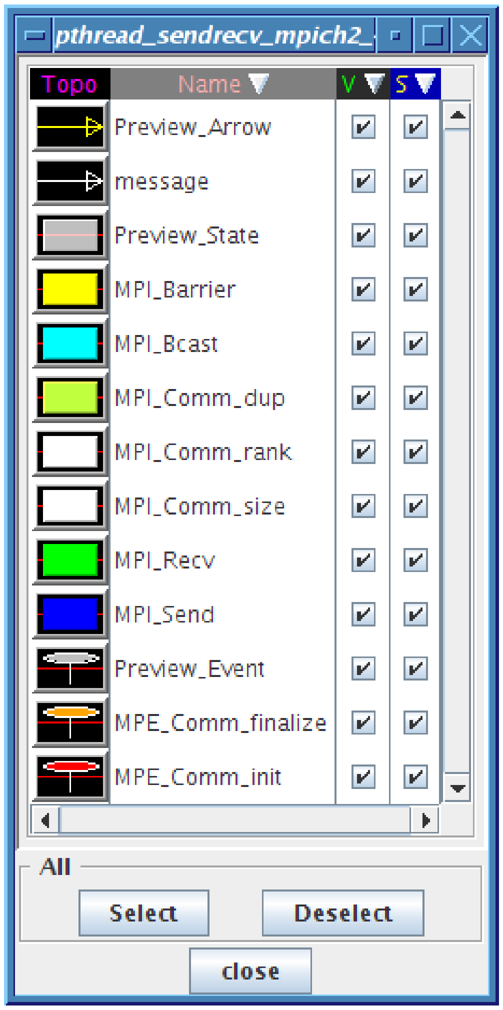
child thread count	MPICH2	OpenMPI+progress_thread	OpenMPI
1	0.025299	0.029545	0.029230
2	0.026213	0.030872	0.032966 4.7
3	0.028916	0.038964	0.050484 4.8
4	0.030145	0.045354	0.054791 4.9
5	0.031977	0.058039	0.149200 4.10
6	0.034462	0.058505	0.193399 4.11

Table 4.2: The total runtime (in second) of the 4-process run of pthread_sendrecv with various number of child threads in different MPI implementations. The 2nd column header, MPICH2: refers to MPICH2-1.0.5p4 built with default sock channel which has MPI_THREAD_MULTIPLE support. The 3rd column header, OpenMPI+progress_thread, refers to OpenMPI-1.2.3 configured with –enable-mpi-threads and –enable-progress-threads. The 4th column, OpenMPI, refers to OpenMPI-1.2.3 built with –enable-mpi-threads which enables the MPI_THREAD_MULTIPLE support.

[4]–enable-threadlogging enables MPE to build a thread-safe MPI logging library which is implemented by using a global mutex over MPE logging library which is not thread-safe yet.

[5]MPE by defaults does –enable-safePMPI to protect the logging code from doing circular logging in unknown MPI implementation where MPI calls are implementated with other MPI calls. Basically, –enable-safePMPI disables the logging before making PMPI call and then re-enables logging when the PMPI call is returned. Using –disable-safePMPI in MPE eliminiates this layer of protection but allows lowest possible logging overhead.

The problematic data in the last column of Table 4.2 are being analyzed with two Jumpshot viewmaps for each run. They are shown in Figures 4.7, 4.8,4.9,4.10 and 4.11. The legend for these pictures are shown in Figure4.6.


pthread_sendrecv_mpich2_

Topo	Name	V	S
→	Preview_Arrow	✓	✓
→	message	✓	✓
⬜	Preview_State	✓	✓
🟨	MPI_Barrier	✓	✓
🟦	MPI_Bcast	✓	✓
🟩	MPI_Comm_dup	✓	✓
⬜	MPI_Comm_rank	✓	✓
⬜	MPI_Comm_size	✓	✓
🟩	MPI_Recv	✓	✓
🟦	MPI_Send	✓	✓
●	Preview_Event	✓	✓
●	MPE_Comm_finalize	✓	✓
●	MPE_Comm_init	✓	✓

Figure 4.6: The legend table of all the pthread_sendrecv runs.

The extra viewmaps provided in MPE logging are:

1) Process-Thread view: where each thread timeline is shown nested under the process timeline it belongs to. Since we are only running 4 processes, only 4 process timelines here.

2) Communicator-Thread view: where each thread is shown nested within the communicator timeline. Since we are runing with 2 to 6 child threads where a duplicated MPI_COMM_WORLD is created for each thread, so we expect to see 3 to 7 major communicator timelines. MPI_COMM_WORLD is always labeled as 0 in CLOG2 converted SLOG-2 file and other duplicated MPI_Comm is labeled with other integer depends on the order of when it is being created.

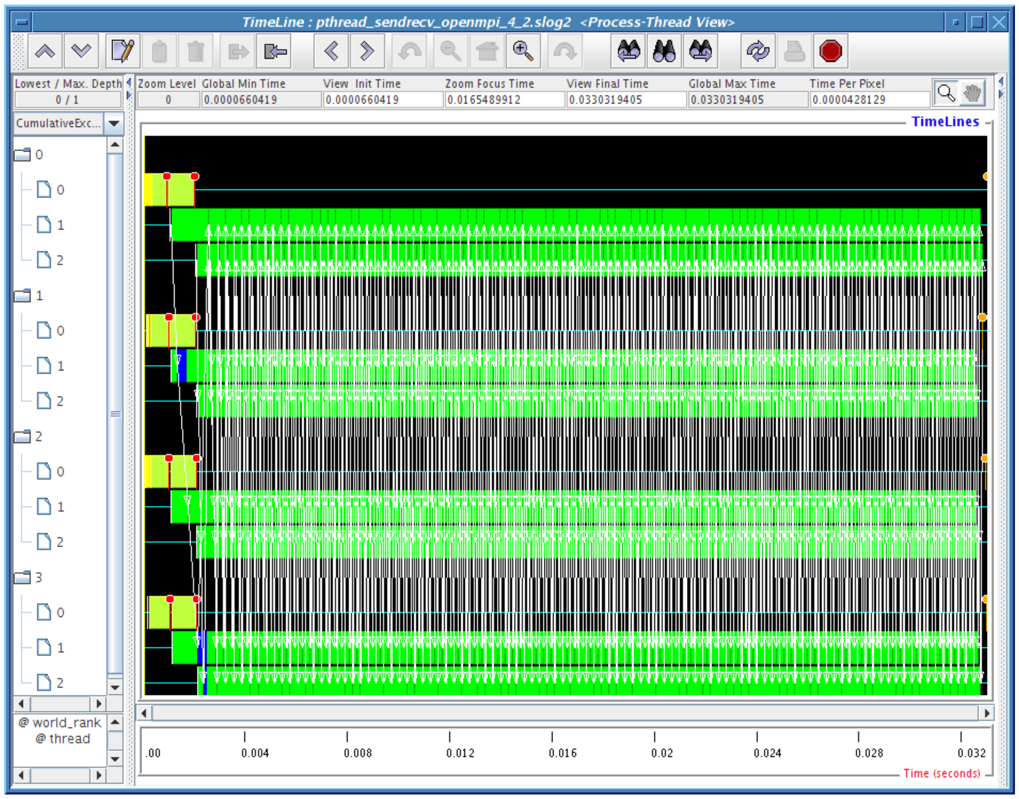
When the timeline window of the process-thread view first shows up, only process timelines are visible, i.e. all the thread timelines are nested within the process timeline. User needs to use the Y-axis LabelExpand button ⇨ or *Alt-E* to expand each process timeline to reveal the thread timeline. Similarly, user can use the Y-axis LabelCollapse button ⇦ or *Alt-C* to collapse the thread timeline back to their corresponding process timeline. Similarly for the communicator-thread view, the Y-axis LabelExpand and LabelCollapse buttons should be used to expand and collapse the communicator timelines.

Figures 4.8, 4.9, 4.10 and 4.11 clearly demonstrate that there is some kind of communication progress problem in OpenMPI when used without progress thread. Without alternating between communicator-thread and process-thread views, it would be difficult to identify the existence of a progress engine problem.

4.4.1 Test program

The following is the test program, pthread_sendrecv.c, used in the previous experiment.

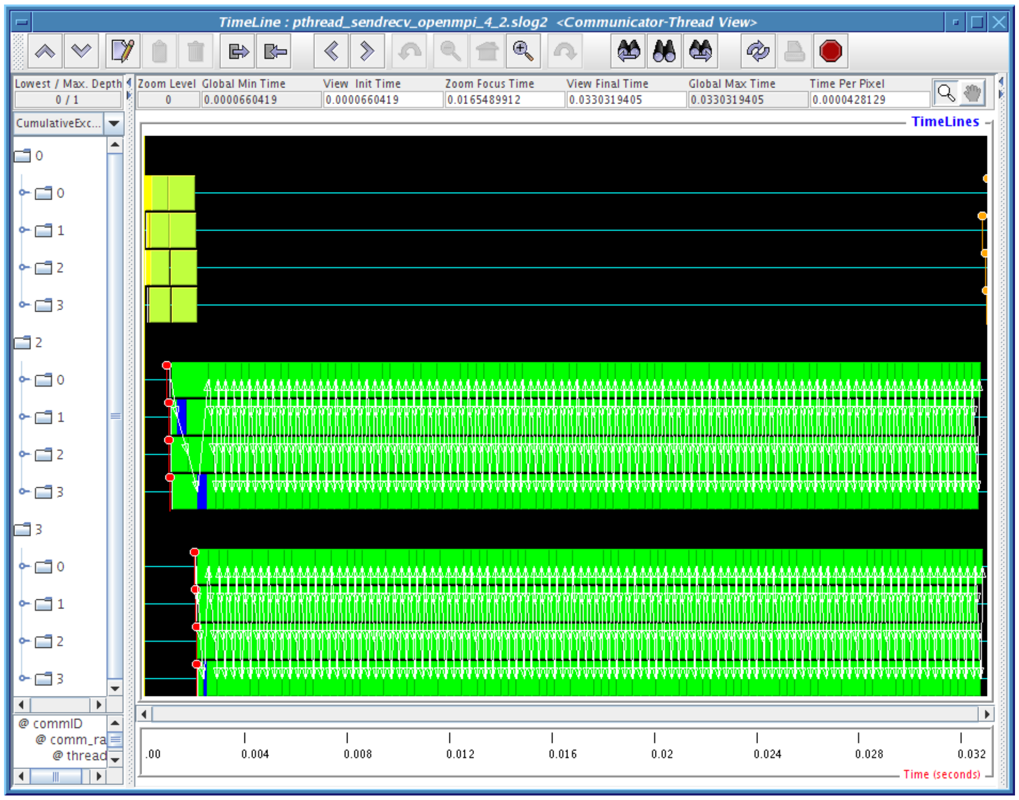
```
/*
   (C) 2007 by Argonne National Laboratory.
       See COPYRIGHT in top-level directory.
*/
#include "mpi.h"
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <string.h>

#define BUFLEN 512
#define NTIMES 100
#define MAX_THREADS 10

/*
    Concurrent send and recv by multiple threads on each process.
*/
void *thd_sendrecv( void * );
void *thd_sendrecv( void *comm_ptr )
{
    MPI_Comm     comm;
    int         my_rank, num_procs, next, buffer_size, namelen, idx;
    char        buffer[BUFLEN], processor_name[MPI_MAX_PROCESSOR_NAME];
    MPI_Status  status;

    comm = *(MPI_Comm *) comm_ptr;

    MPI_Comm_size( comm, &num_procs );
    MPI_Comm_rank( comm, &my_rank );
    MPI_Get_processor_name( processor_name, &namelen );

    fprintf( stderr, "Process %d on %s\n", my_rank, processor_name );
    strcpy( buffer, "hello there" );
    buffer_size = strlen(buffer)+1;
```



TimeLine : pthread_sendrecv_openmpi_4_2.slog2 <Process-Thread View>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 1	0	0.0000660419	0.0000660419	0.0165489912	0.0330319405	0.0330319405	0.0000428129

(a) process-thread view

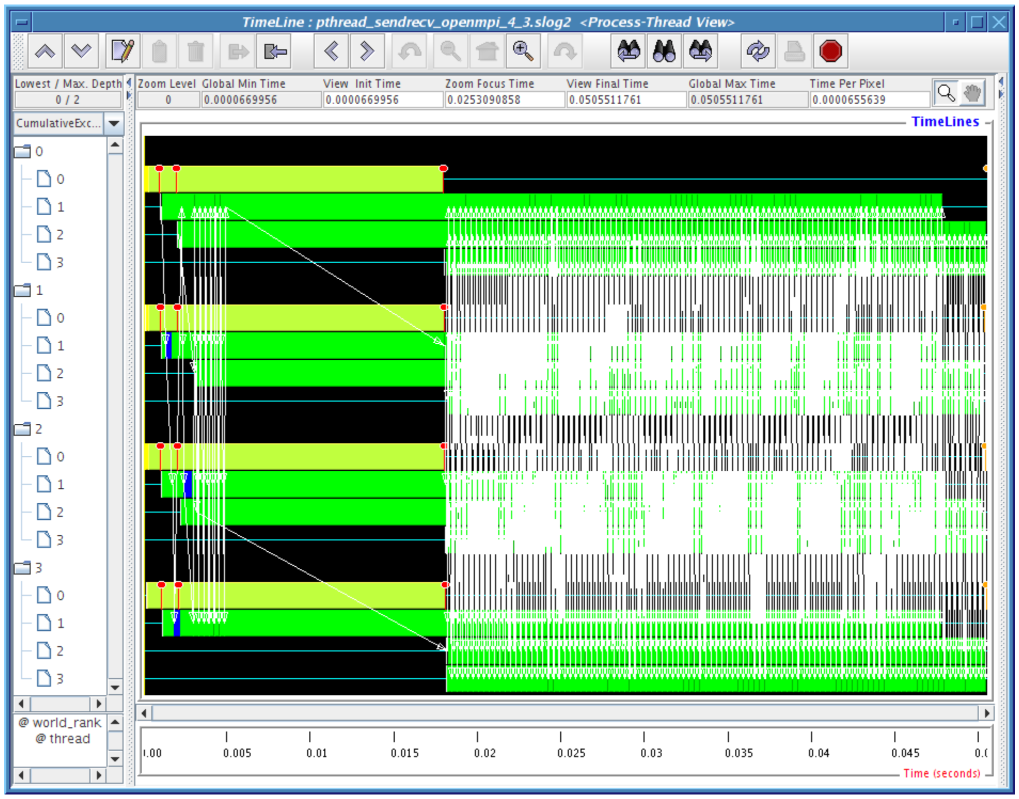


TimeLine : pthread_sendrecv_openmpi_4_2.slog2 <Communicator-Thread View>

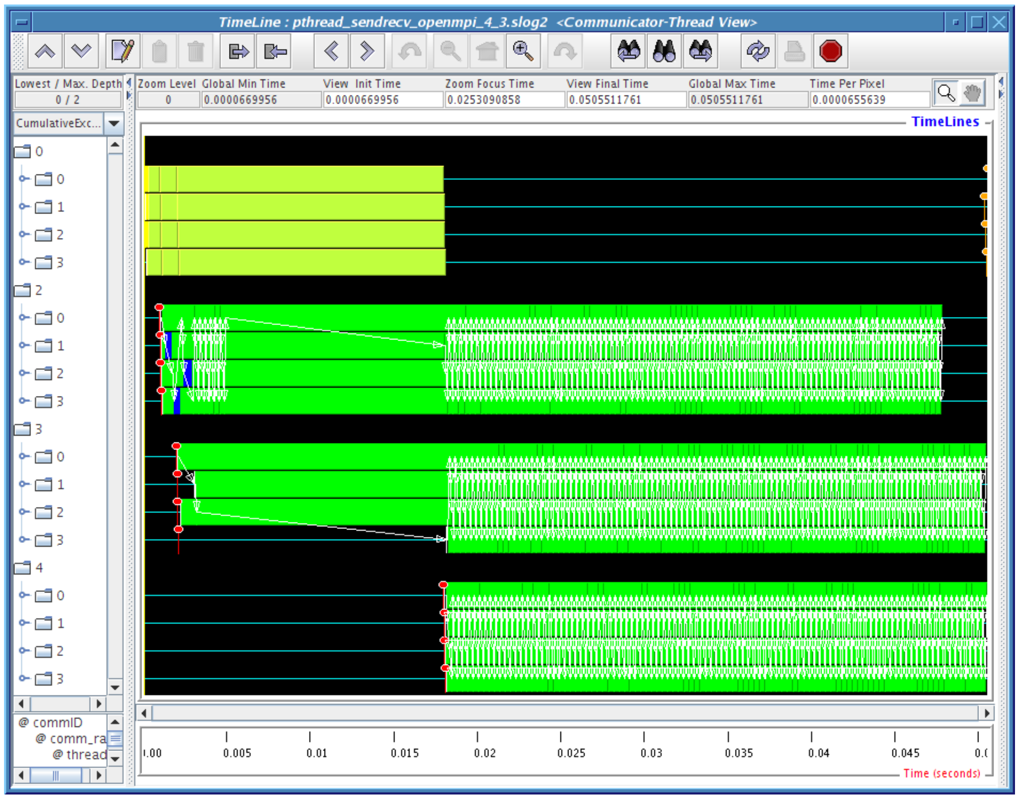
Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 1	0	0.0000660419	0.0000660419	0.0165489912	0.0330319405	0.0330319405	0.0000428129

(b) communicator-thread view

Figure 4.7: OpenMPI without progress thread: 2 child threads per process. As shown in both the Process-Thread view and Communicator-Thread views here, everything performs as expected.

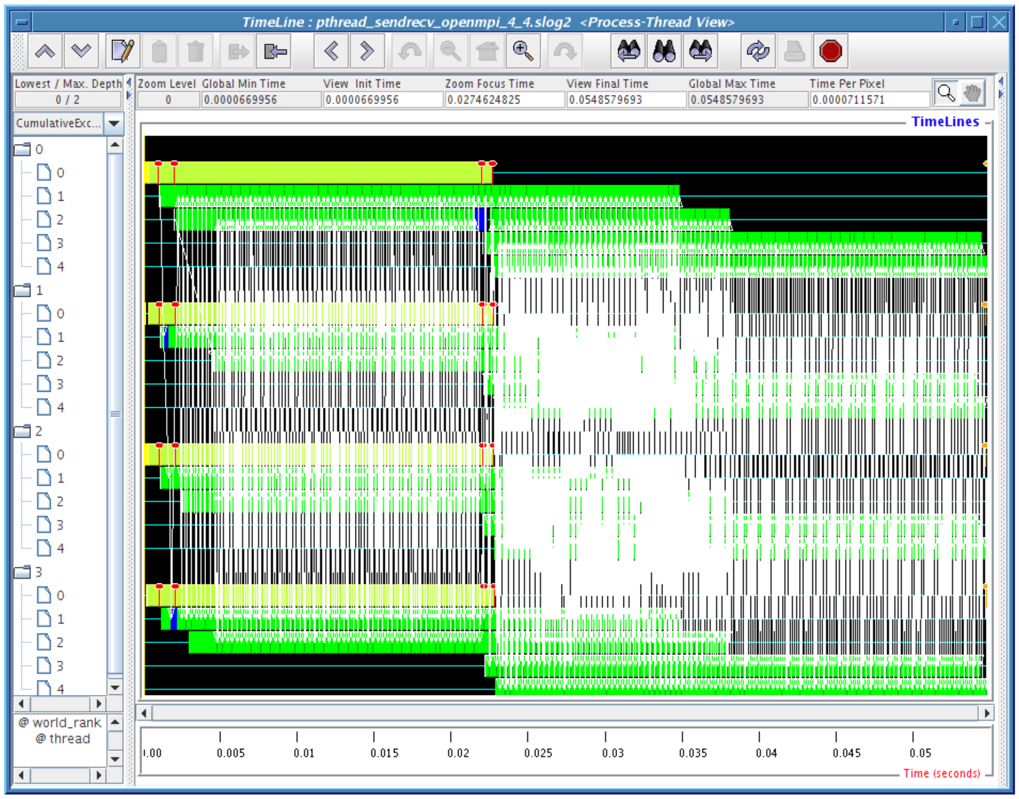


(a) process-thread view



(b) communicator-thread view

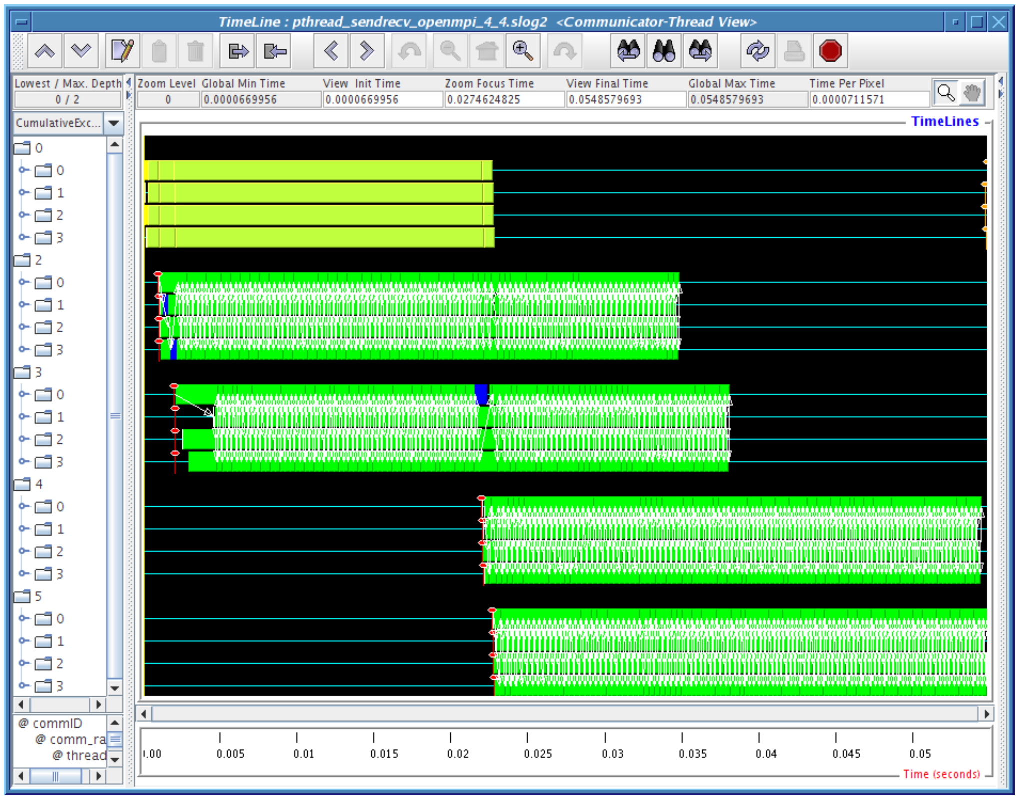
Figure 4.8: OpenMPI without progress thread: 3 child threads per process where there are 3 MPI_Comm_dup() calls in the master thread 0. As shown in the expanded Process-Thread view, the 3rd MPI_Comm_dup() call takes significantly longer than the first two MPI_Comm_dup(). The expanded Communicator-Thread view also suggests that the delayed 3rd MPI_Comm_dup() is blocking MPI point-to-point communication in the first two duplicated MPI_COMM_WORLD. As soon as the delayed MPI_Comm_dup() exits, the MPI point-to-point communication is restored.


TimeLine : pthread_sendrecv_openmpi_4_4.slog2 <Process-Thread View>

Lowest / Max. Depth: 0 / 2
Zoom Level: 0
Global Min Time: 0.0000669956
View Init Time: 0.0000669956
Zoom Focus Time: 0.0274624825
View Final Time: 0.0548579693
Global Max Time: 0.0548579693
Time Per Pixel: 0.0000711571

CumulativeExc...

(a) process-thread view

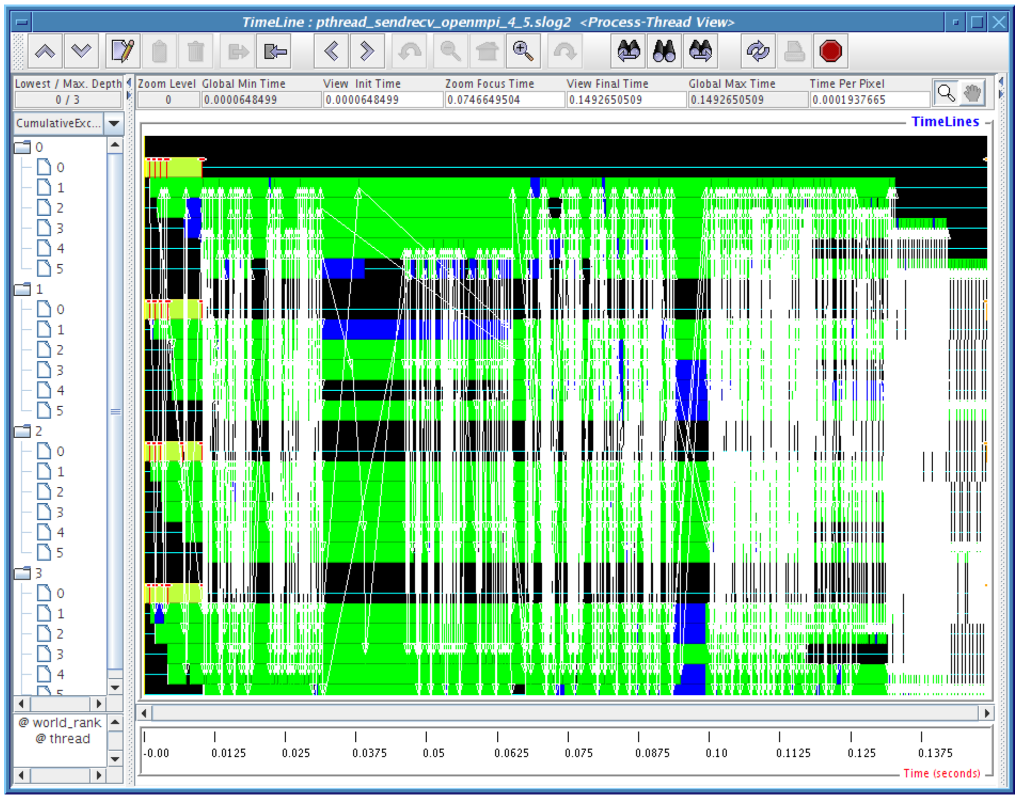

TimeLine : pthread_sendrecv_openmpi_4_4.slog2 <Communicator-Thread View>

Lowest / Max. Depth: 0 / 2
Zoom Level: 0
Global Min Time: 0.0000669956
View Init Time: 0.0000669956
Zoom Focus Time: 0.0274624825
View Final Time: 0.0548579693
Global Max Time: 0.0548579693
Time Per Pixel: 0.0000711571

CumulativeExc...

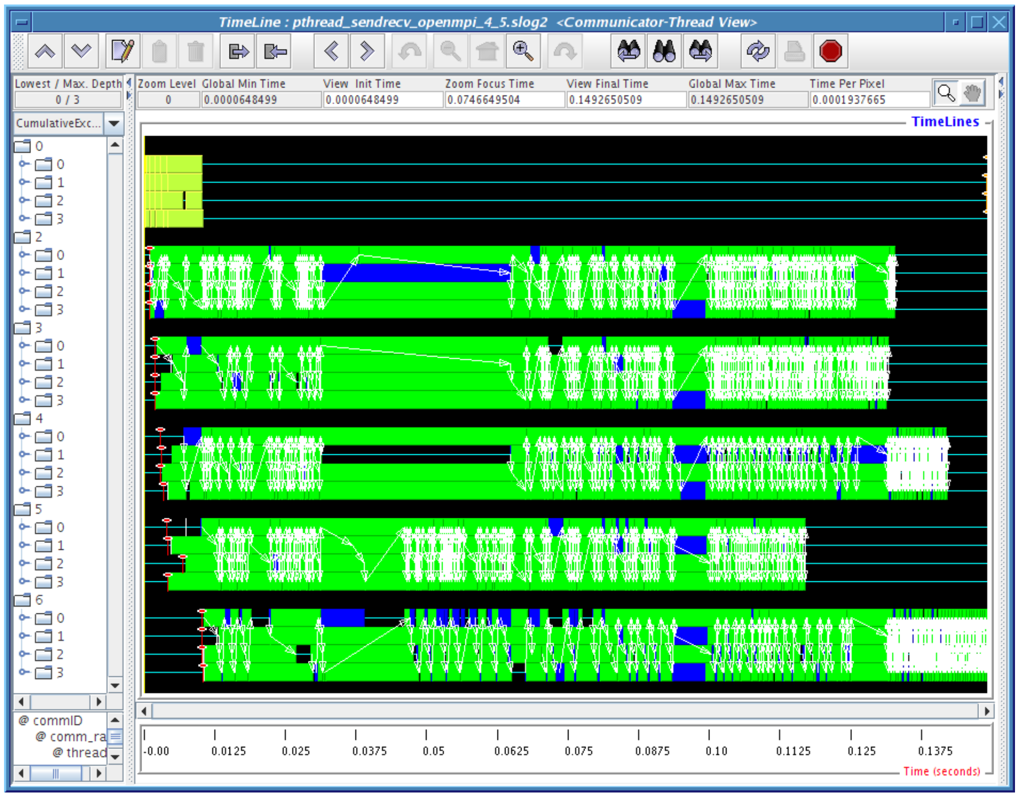
(b) communicator-thread view

Figure 4.9: OpenMPI without progress thread: 4 child threads per process. Similar to Fig. 4.8, the 3rd MPI_Comm_dup() is delayed but not the 4th MPI_Comm_dup(). The interference between the delayed 3rd MPI_Comm_dup() and the other dup MPI_COMM_WORLD seen in Fig. 4.8 is not observed here. So the communication in first two dup MPI_COMM_WORLD finishes much earlier than the communication in the last two communicators.


TimeLine : pthread_sendrecv_openmpi_4_5.slog2 <Process-Thread View>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 3	0	0.0000648499	0.0000648499	0.0746649504	0.1492650509	0.1492650509	0.0001937665

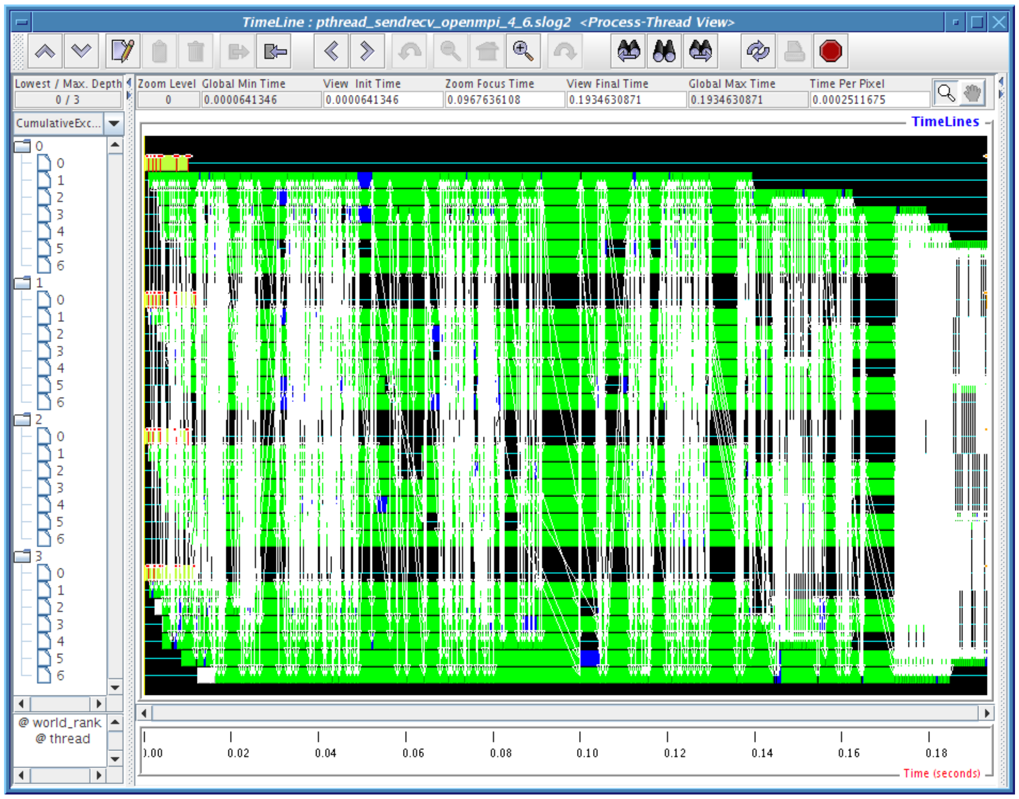
(a) process-thread view


TimeLine : pthread_sendrecv_openmpi_4_5.slog2 <Communicator-Thread View>

Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 3	0	0.0000648499	0.0000648499	0.0746649504	0.1492650509	0.1492650509	0.0001937665

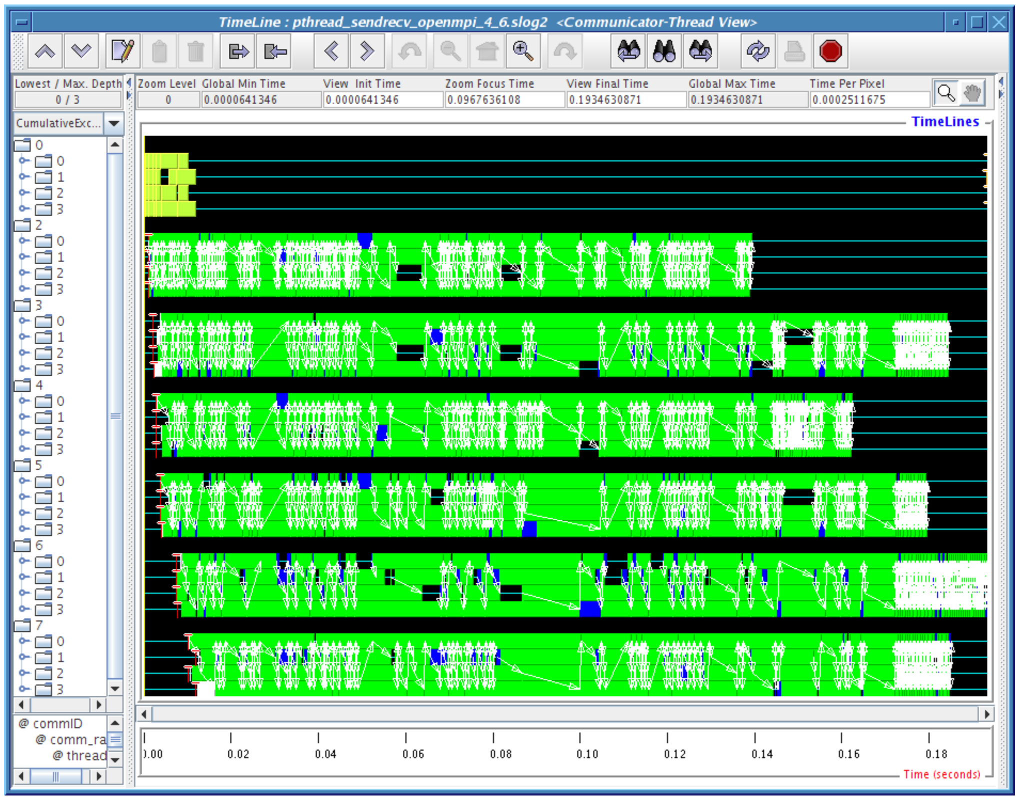
(b) communicator-thread view

Figure 4.10: OpenMPI without progress thread: 5 child threads per process. Again, the last MPI_Comm_dup() takes longer than previous MPI_Comm_dup()s in finishing up. The feature that we observed in Fig. 4.8 that the delayed MPI_Comm_dup() is blocking other communicator's communication occurs here. However, even long after all MPI_Comm_dup() are done, there are many regions in the communicator-thread view that MPI communication is not progressing, i.e. some kind of temporary deadlock in the MPI progress engine may be happening here.



Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 3	0	0.0000641346	0.0000641346	0.0967636108	0.1934630871	0.1934630871	0.0002511675

(a) process-thread view



Lowest / Max. Depth	Zoom Level	Global Min Time	View Init Time	Zoom Focus Time	View Final Time	Global Max Time	Time Per Pixel
0 / 3	0	0.0000641346	0.0000641346	0.0967636108	0.1934630871	0.1934630871	0.0002511675

(b) communicator-thread view

Figure 4.11: OpenMPI without progress thread: 6 child threads per process. This is very similar to Fig. 4.10.

```
    if ( my_rank == num_procs-1 )
        next = 0;
    else
        next = my_rank+1;

    for ( idx = 0; idx < NTIMES; idx++ ) {
        if (my_rank == 0) {
            MPI_Send(buffer, buffer_size, MPI_CHAR, next, 99, comm);
            MPI_Send(buffer, buffer_size, MPI_CHAR, MPI_PROC_NULL, 299, comm);
            MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99,
                     comm, &status);
        }
        else {
            MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_ANY_SOURCE, 99,
                     comm, &status);
            MPI_Recv(buffer, BUFLEN, MPI_CHAR, MPI_PROC_NULL, 299,
                     comm, &status);
            MPI_Send(buffer, buffer_size, MPI_CHAR, next, 99, comm);
        }
        /* MPI_Barrier(comm); */
    }

    pthread_exit( NULL );
    return 0;
}



int main( int argc,char *argv[] )
{
    MPI_Comm   comm[ MAX_THREADS ];
    pthread_t  thd_id[ MAX_THREADS ];
    int        my_rank, ii, provided;
    int        num_threads;

    MPI_Init_thread( &argc, &argv, MPI_THREAD_MULTIPLE, &provided );
    if ( provided != MPI_THREAD_MULTIPLE ) {
        printf( "Aborting, MPI_THREAD_MULTIPLE is needed...\n" );
        MPI_Abort( MPI_COMM_WORLD, 1 );
    }

    MPI_Comm_rank( MPI_COMM_WORLD, &my_rank );

    if ( my_rank == 0 ) {
        if (argc != 2) {
            printf( "Error: %s num_threads\n", argv[0] );
            MPI_Abort( MPI_COMM_WORLD, 1 );
        }
        num_threads = atoi( argv[1] );
        MPI_Bcast( &num_threads, 1, MPI_INT, 0, MPI_COMM_WORLD );
    }
    else
        MPI_Bcast( &num_threads, 1, MPI_INT, 0, MPI_COMM_WORLD );

    MPI_Barrier( MPI_COMM_WORLD );

    for ( ii=0; ii < num_threads; ii++ ) {
        MPI_Comm_dup( MPI_COMM_WORLD, &comm[ii] );
        pthread_create( &thd_id[ii], NULL, thd_sendrecv, (void *) &comm[ii] );
    }

    for ( ii=0; ii < num_threads; ii++ )
        pthread_join( thd_id[ii], NULL );

    MPI_Finalize();
    return 0;
}
```