Oracle® Solaris Studio 12.2: Performance Analyzer MPI Tutorial

Part No: 820–5551
September 2010

ORACLE®

110606@25097

Contents

Preface

This *Performance Analyzer MPI Tutorial* provides step-by-step instructions for collecting an experiment on an MPI program and using the Performance Analyzer to examine MPI data in the experiment.

Who Should Use This Book

This document is intended for application developers with a working knowledge of C, C++, or Fortran programming languages and the Message Passing Interface (MPI) specification. The users of this document need some understanding of the Solaris operating system, or the Linux operating system, and UNIX operating system commands. Some knowledge of performance analysis is helpful but is not required.

Supported Platforms

This Oracle Solaris Studio release supports systems that use the SPARC and x86 families of processor architectures: UltraSPARC, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems for the version of the Solaris Operating System you are running are available in the hardware compatibility lists at `http://www.sun.com/bigadmin/hcl`. These documents cite any implementation differences between the platform types.

In this document, these x86 related terms mean the following:

- “x86” refers to the larger family of 64-bit and 32-bit x86 compatible products.
- ”x64” points out specific 64-bit information about AMD64 or EM64T systems.
- “32-bit x86” points out specific 32-bit information about x86 based systems.

For supported systems, see the hardware compatibility lists.

Related Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Accessing Oracle Solaris Studio Documentation

You can access the documentation at the following locations:

- The documentation is available from the documentation index page at `http://www.oracle.com/technetwork/server-storage/solarisstudio/documentation/index.html`.
- Online help for the IDE is available through the Help menu, as well as through the F1 key and through Help buttons on many windows and dialog boxes, in the IDE.
- Online help for the Performance Analyzer and the Thread Analyzer is available through the Help menu, as well as through the F1 key and through Help buttons on many windows and dialog boxes in these tools.
- Online help for dbxtool and DLight is available through the Help menu, as well as through the F1 key and through Help buttons on many dialog boxes in these tools.

Documentation in Accessible Formats

The documentation is provided in accessible formats that are readable by assistive technologies for users with disabilities. You can find accessible versions of documentation as described in the following table. If your software is not installed in the `/opt` directory, ask your system administrator for the equivalent path on your system.

Type of Documentation	Format and Location of Accessible Version
Manuals	HTML from the Oracle Solaris Studio 12.2 collection on docs.sun.com
What's New in the Oracle Solaris Studio 12.2 Release (Information that was included in the component Readmes in previous releases)	HTML from the Oracle Solaris Studio 12.2 collection on docs.sun.com

Type of Documentation	Format and Location of Accessible Version
Man pages	In the installed product through the man command
Online help	HTML through the Help menu, Help buttons, and the F1 key in the IDE, dbxtool, DLight, and the Performance Analyzer
Release notes	HTML from the Oracle Solaris Studio 12.2 collection on docs.sun.com

Documentation, Support, and Training

See the following web sites for additional resources:

- Documentation (http://docs.sun.com)
- Support (http://www.oracle.com/us/support/systems/index.html)
- Training (http://education.oracle.com) – Click the Sun link in the left navigation bar.

Oracle Welcomes Your Comments

Oracle welcomes your comments and suggestions on the quality and usefulness of its documentation. If you find any errors or have any other suggestions for improvement, go to http://docs.sun.com and click Feedback. Indicate the title and part number of the documentation along with the chapter, section, and page number, if available. Please let us know if you want a reply.

Oracle Technology Network (http://www.oracle.com/technetwork/index.html) offers a range of resources related to Oracle software:

- Discuss technical problems and solutions on the Discussion Forums (http://forums.oracle.com).
- Get hands-on step-by-step tutorials with Oracle By Example (http://www.oracle.com/technology/obe/start/index.html).
- Download Sample Code (http://www.oracle.com/technology/sample_code/index.html).

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

CHAPTER 1

Performance Analyzer MPI Tutorial

This tutorial is designed to be followed from beginning to end to show you how to use the Oracle Solaris Studio 12.2 Performance Analyzer's MPI features. The tutorial uses a sample program to demonstrate how to use the MPI Timeline and MPI Chart tabs.

About MPI and Performance Analyzer

You can use the Performance Analyzer to examine Message Passing Interface (MPI) applications to answer the following questions:

- Would tuning the MPI code produce significant performance improvement?
- Is the MPI performance characterized by synchronization or data transfer?
- Does the program contain load imbalances?
- How long is one iteration of program execution?
- How long does it take for program performance to equilibrate?
- What are the message-passing patterns in program execution?
- Which are most important: long or short messages?
- Do processes that send messages synchronize with processes that receive messages?

While the preceding list is too broad to address in a single document, this tutorial guides you through some new features of the Performance Analyzer including the following:

- MPI Timeline. A graphical display of the MPI activity that occurred during an application's run.
- MPI Chart. A tool that generates scatter plots and histograms to visualize the performance data of MPI functions and MPI messages.
- MPI data-zooming and data-filtering. A set of controls that you can use to broaden or narrow your view of the data in the MPI Timeline and MPI Chart.

The MPI Timeline tab presents the data from a run of the test program as a timeline. Initially, your view of the timeline encompasses the run from beginning to end with all MPI functions and MPI messages represented graphically in a condensed form. You'll learn how to expand this presentation and move down from a complete view to a tightly focused view that can be as granular as a single function. The MPI Timeline tab offers many different ways to zoom, pan, and examine the data, together with MPI Chart tab. The MPI Chart tab enables you to plot statistical data about the functions and messages in graphical charts, to help you see what is happening in the run.

See the manual *Oracle Solaris Studio 12.2: Performance Analyzer* for detailed information about the Performance Analyzer.

Setting Up for the Tutorial

The Performance Analyzer works with several implementations of the Message Passing Interface (MPI) standard, including the Oracle Message Passing Toolkit, a highly optimized implementation of MPI for Sun x86 and SPARC-based systems. The Oracle Message Passing Toolkit, formerly Sun HPC ClusterTools, must be at least version 7.

This tutorial explains how to use the Performance Analyzer on an example MPI application called `ring_c`, which is included with the Oracle Message Passing Toolkit. You must already have a cluster configured and functioning for this tutorial.

Follow the steps below to get started.

1. Download the Oracle Message Passing Toolkit 8.1.2c from `http://www.oracle.com/us/products/tools/message-passing-toolkit-070499.html`.
2. Install the toolkit software as described in the *Sun HPC ClusterTools 8.2.1c Software Installation Guide*, which is available as a PDF download at `http://dlc.sun.com/pdf/821-1318-10/821-1318-10.pdf`.
3. Add the /*Studio-installation-directory*/`bin` directory and the *OracleMessagePassingToolkit-installation-directory*/`bin` directory to your path.

   On Solaris systems, the default paths are `/opt/solstudio12.2/bin` and `/opt/SUNWhpc/HPC8.2.1c/sun/bin`.

   On Linux systems, the default paths are `/opt/oracle/solstudio12.2/bin` and `/opt/SUNWhpc/HPC8.2.1c/sun/bin`.
4. Copy the /*OracleMessagePassingToolkit-installation-directory*/`examples` directory into a directory to which you have write access. The newly copied `examples` directory must be visible from all the cluster nodes.
5. Change directory to your new `examples` directory.
6. Build the `ring_c` example.

```
% make ring_c
mpicc -g -o ring_c ring_c.c
```

The program is compiled with the -g option, which allows the Performance Analyzer data collector to map MPI events to source code.

7. Run the ring_c example with mpirun to make sure it works correctly. The ring_c program simply passes a message from process to process in a ring, then terminates.

   This example shows how to run the program on a two-node cluster. The node names are specified in a host file, along with the number of slots that are to be used on each node. The tutorial uses 25 processes, and specifies one slot on each host. You should specify a number of processes and slots that is appropriate for your system. See the mpirun(1) man page for more information about specifying hosts and slots. You can also run this command on a standalone host that isn't part of a cluster, but the results might be less educational.

   The host file for this example is called clusterhosts and contains the following content:

   ```
   hostA slots=1
   hostB slots=1
   ```

   You must have permission to use a remote shell (ssh/rsh) to each host without logging into the hosts. By default, mpirun uses ssh.

   ```
   % mpirun -np 25 --hostfile clusterhosts ring_c
   Process 0 sending 10 to 1, tag 201 (25 processes in ring)
   Process 0 sent to 1
   Process 0 decremented value: 9
   Process 0 decremented value: 8
   Process 0 decremented value: 7

   Process 0 decremented value: 6
   Process 0 decremented value: 5
   Process 0 decremented value: 4
   Process 0 decremented value: 3
   Process 0 decremented value: 2
   Process 0 decremented value: 1

   Process 0 decremented value: 0
   Process 0 exiting
   Process 1 exiting
   Process 2 exiting
   .
   .
   .
   Process 24 exiting
   ```

   Run this command and if you get similar output, you are ready to collect data on an example application as shown in the next section.

If you have problems with mpirun using ssh, try using the option --mca plm_rsh_agent rsh with the mpirun command to connect using the rsh command:

```
% mpirun -np 25 --hostfile clusterhosts --mca plm_rsh_agent rsh -- ring_c
```

Collecting Data on the ring_c Example

1. Change to the directory where your example binaries and source code are located.
2. Run the following command:

   ```
   % collect -M OMPT mpirun -np 25 --hostfile clusterhosts  -- ring_c
   ```

   The command might take a few moments to run and the output should be the same as the test run through the mpirun command.

   The -M OMPT option indicates the MPI version is the Oracle Message Passing Toolkit. See the collect(1) man page for more information about MPI versions supported.

   The -np 25 option specifies 25 processes on the cluster, and --hostfile clusterhosts indicates that the node names and the number of slots that are to be used on each node are specified in a file called clusterhosts.

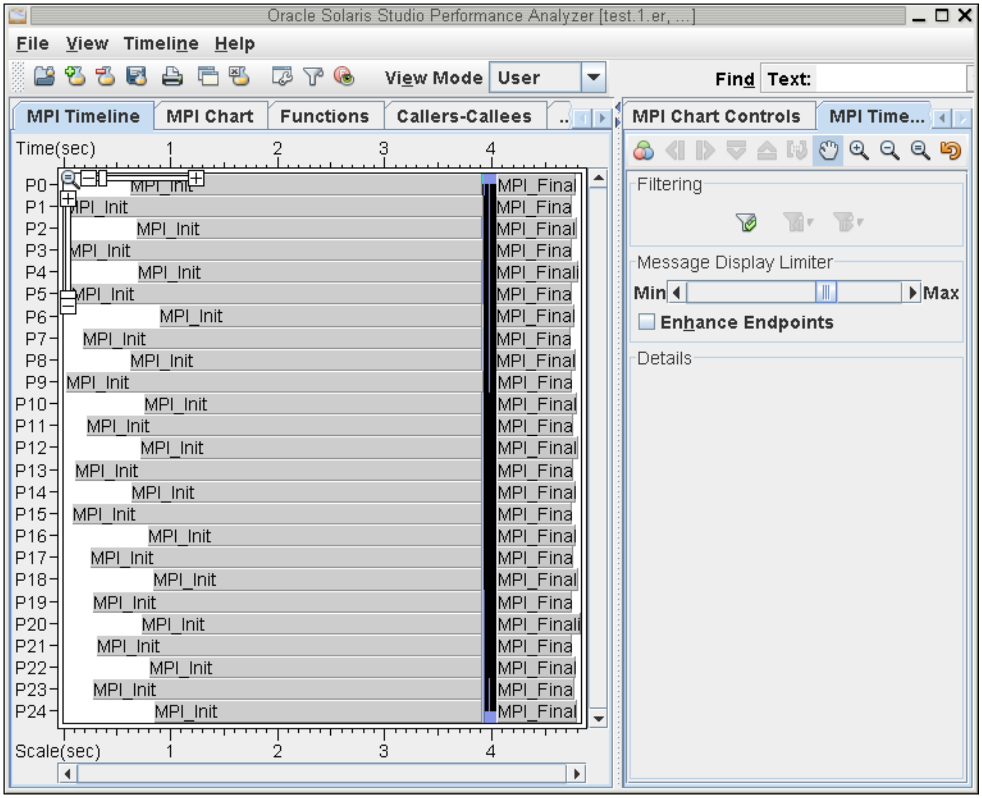
   This command specifies to use 25 processes on two hosts, and specifies one slot on each host. You should specify a number of processes and slots that is appropriate for your system.
3. List the contents of the newly created test.1.er directory and make sure the date on the files reflects the latest execution. This means you ran the command successfully and are ready to run the Performance Analyzer on ring_c. The integer in test.1.er increments for each collect command you run so the rest of this tutorial refers to this name generically as test.*.er.

Opening the Experiment

1. Change to the directory that contains the ring_c.c source file, the ring_c executable, and the test.*.er directory.
2. Start the Performance Analyzer from the command line:

   ```
   %  analyzer
   ```

   The Performance Analyzer opens a file browser for you to find and open an experiment. If not, choose File > Open Experiment.
3. Find the test.*.er experiment that you just created and open it. The Performance Analyzer window should look similar to that below.



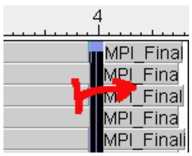
The experiment opens on the MPI Timeline tab. The MPI Chart tab is next to it. In the right panel you can see the MPI Chart Controls and MPI Timeline Controls tabs.

The MPI Timeline shows a view of the data over time as the program was run through the collector. The horizontal axis shows elapsed time. At the bottom, the horizontal axis shows *relative time* with the origin at the left edge of the display. At the top, the horizontal axis shows *absolute time* where the origin is the start of the data. The vertical axis shows MPI process rank. For each MPI process you can look horizontally to see what the process is doing as a function of elapsed time.

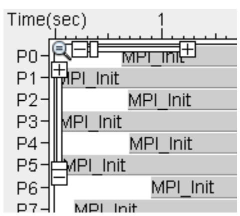
This initial view of the timeline answers the question: What is the time scale of program execution? The screen capture shows the time scale is approximately 5 seconds. However, the actual run time spans 3.90 to 4.05 seconds, the steady state of the application program. The `collect` tool uses `MPI_Init` and `MPI_Finalize` to set up and terminate data collection.

Navigating the MPI Timeline

1. Click the MPI Timeline tab if it is not already selected.
2. Zoom in on the data by clicking and dragging from the left to the right on any process row as shown by the directional arrow in the graphic below. When you release the mouse button, the area inside the box automatically expands to reveal a zoomed in view.

   An alternative to clicking and dragging is to use the zooming slider controls in the top left of the timeline.

   Use the horizontal slider to change the time scale. You'll see progressively smaller chunks of time, while still showing all the processes, as you zoom.

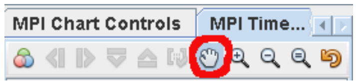
   Use the vertical slider to zoom in on the MPI processes.

   Click the Zoom Undo button in the MPI Timeline Controls tab shown below to go back to the previous level of zooming.

   Click the Zoom Undo button a second time to return to the first zoom.
3. Pan across the data by sliding the scroll bars located at the bottom and the right of the timeline.

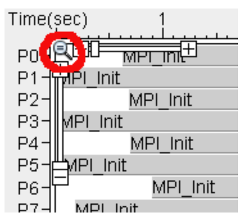
Alternatively you can toggle between a pointer that zooms and a pointer that pans by clicking the hand icon in the MPI Timeline Controls tab.



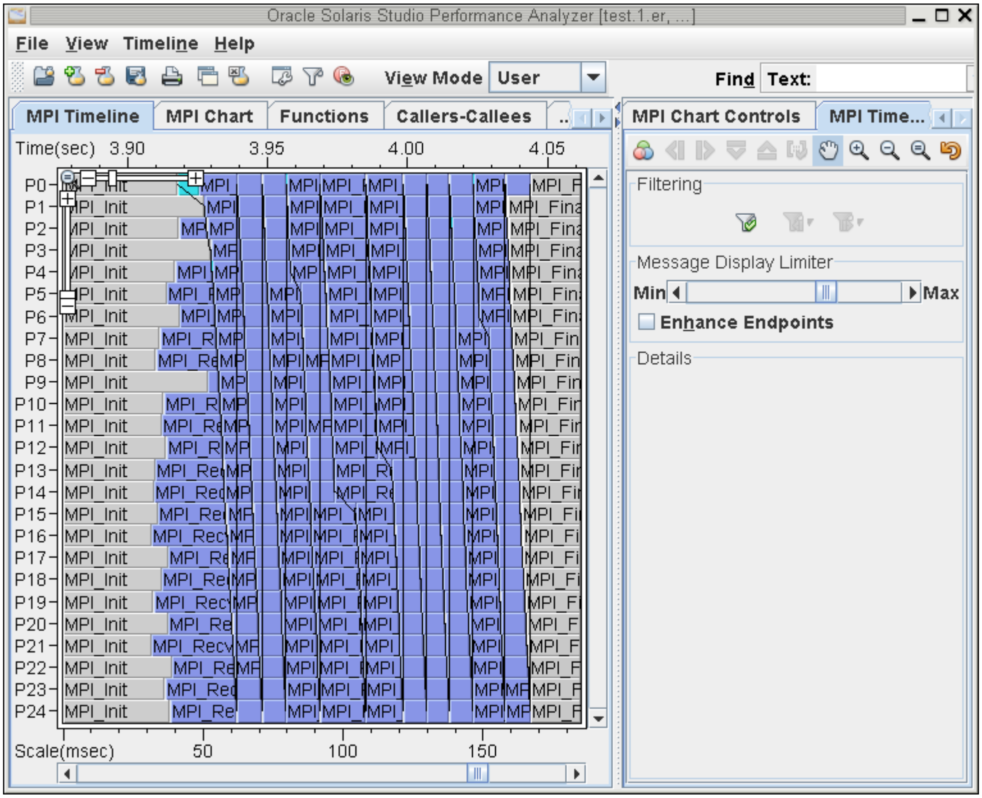
When the pointer is a hand, you can drag across the MPI Timeline to pan horizontally.

Viewing Message Details

1. Reset the view to the original, maximum, zoomed-out setting by clicking the Zoom Reset button, which is located to the top left of the zoom sliders.



2. Zoom in on the activity area by dragging on the area horizontally with the mouse so it looks similar to what you see here.



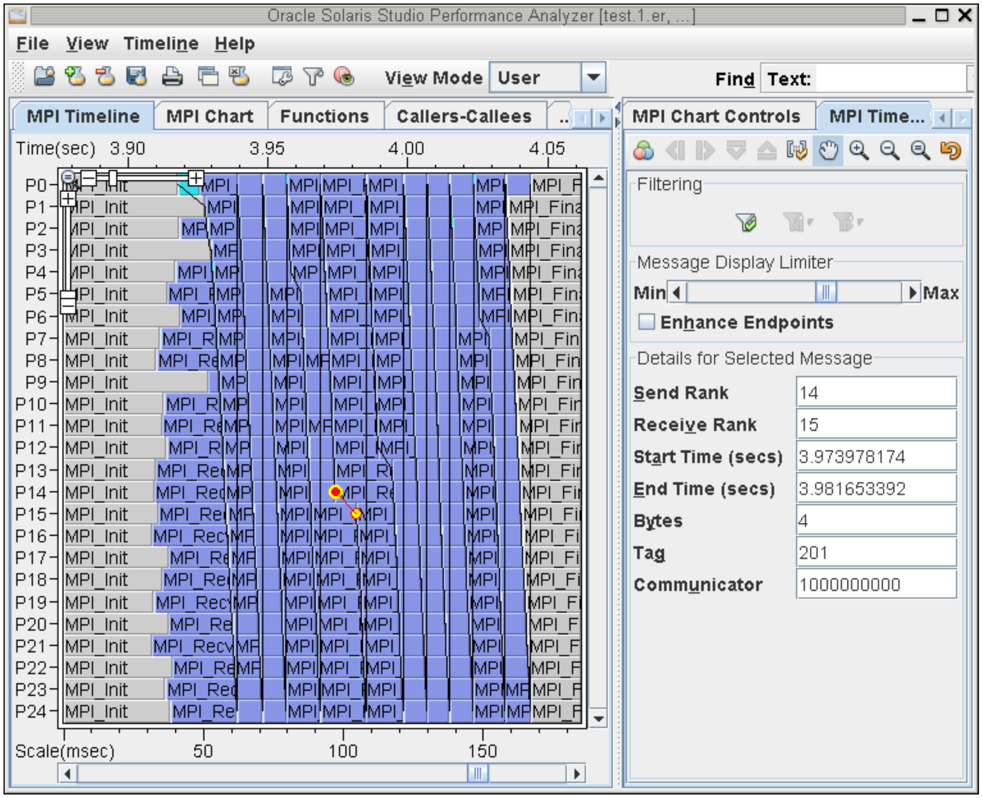
In the zoomed in timeline, now you can see that the steady state portion of the program execution appears to be from 3.93 seconds to 4.03 seconds.

You can also see that MPI functions are color coded. The black lines drawn between events represent point-to-point messages exchanged by the MPI processes.

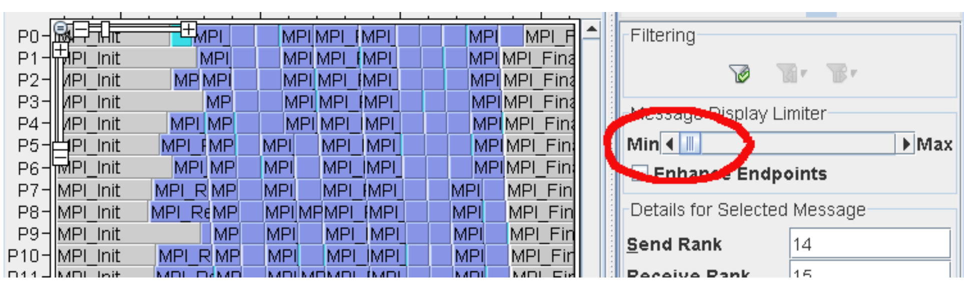
With this view of the timeline, you can answer the question: How long is one iteration before the pattern repeats? The answer is roughly 10 milliseconds. Look at the relative time scale at the bottom to see how often the loop seems to repeat.

3. Click one of the black message lines.

   The line turns red and details about the message are displayed in the right-hand panel MPI Timeline Controls tab.



4. In the MPI Timeline Controls tab, find the Messages slider, then click and drag it to Min as shown.



The Messages slider controls the number of message lines displayed on the screen. At Min, only functions are displayed in the MPI Timeline tab.

In this simple example, all of the messages can be displayed. However, displaying all messages in complex applications can overwhelm the tool and make the screen too cluttered to be usable. Select a lower limit to reduce the number of messages shown in the timeline. If fewer than 100% of the messages are shown, the messages used are those messages that are most costly in terms of the total time used in the message's send and receive functions.

5. Set the Messages slider back to Max.

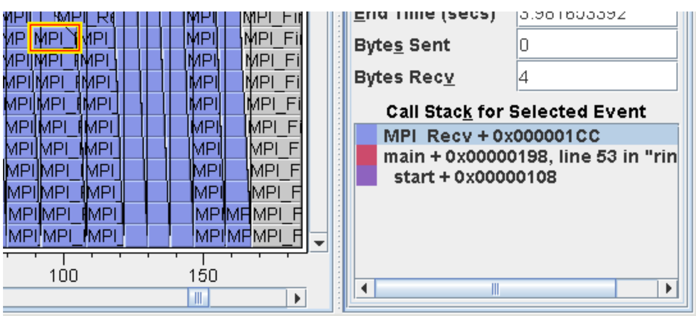
Viewing Function Details and Application Source Code

1. Click on one of the `MPI_Recv` function events in the MPI Timeline tab.

   The function is highlighted in yellow, and details about the function are displayed in the MPI Timeline Controls tab on the right.

2. In the MPI Timeline Controls tab, click the button labeled Show Call Stack if available.

   After a few moments, the call stack for the highlighted state should be shown in the MPI Timeline Controls tab:

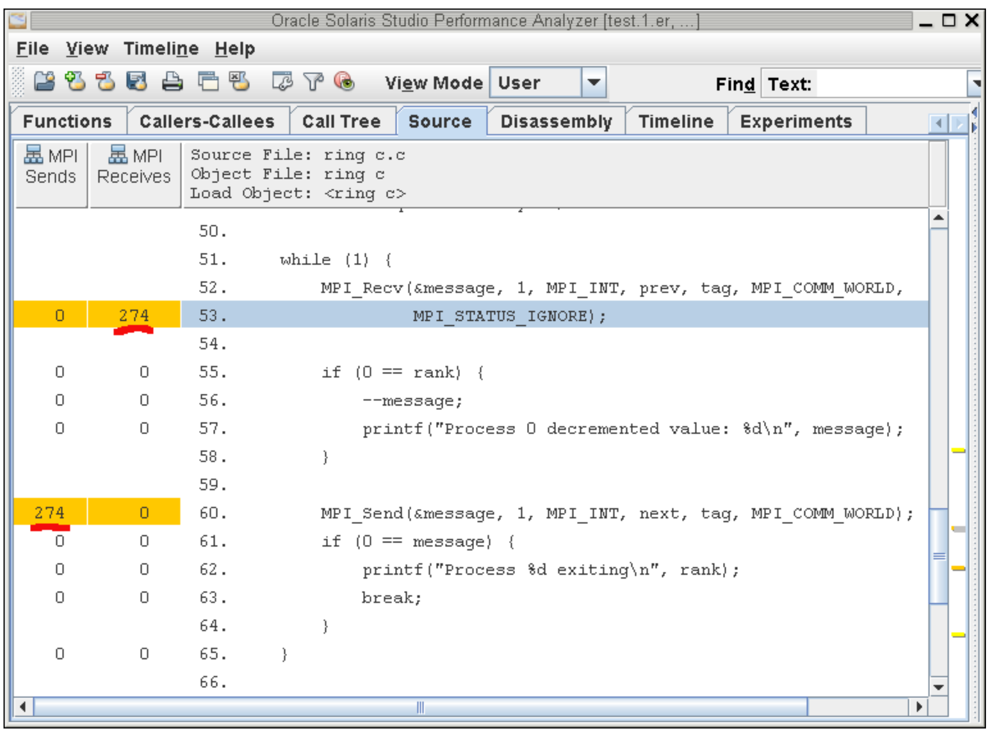


3. When the Call Stack for Selected Event is displayed in the MPI Timeline Controls tab, click on `main + 0x00000198, line 53 in "ring_c.c`.
4. Click the Source tab in the main Performance Analyzer panel.

If you get a message such as `Object file (unknown) not readable`, make sure you selected the stack frame `main + 0x00000198, line 53 in "ring_c.c`.

Note – Source is only visible when the source is in the same location it was in when the program was run through the collector, or when it can be found in the `$expts` path as set in `.er.rc` or in View > Set Data Presentation > Search Path. Source also needs to be compiled with `-g`. If the source code is not visible, you may not have started the Analyzer from the directory containing the `ring_c` binary and source code. If this is the case, quit the Performance Analyzer and restart after you change to the directory containing `ring_c`.

When the source is visible, it should show where `main()` calls the `MPI_Recv()` function. As shown below, `MPI_Recv()` is called from line 53 in the source. The metrics with high values are highlighted: 274 receives are associated with line 53, and 274 sends are associated with `MPI_Send()` on line 60.

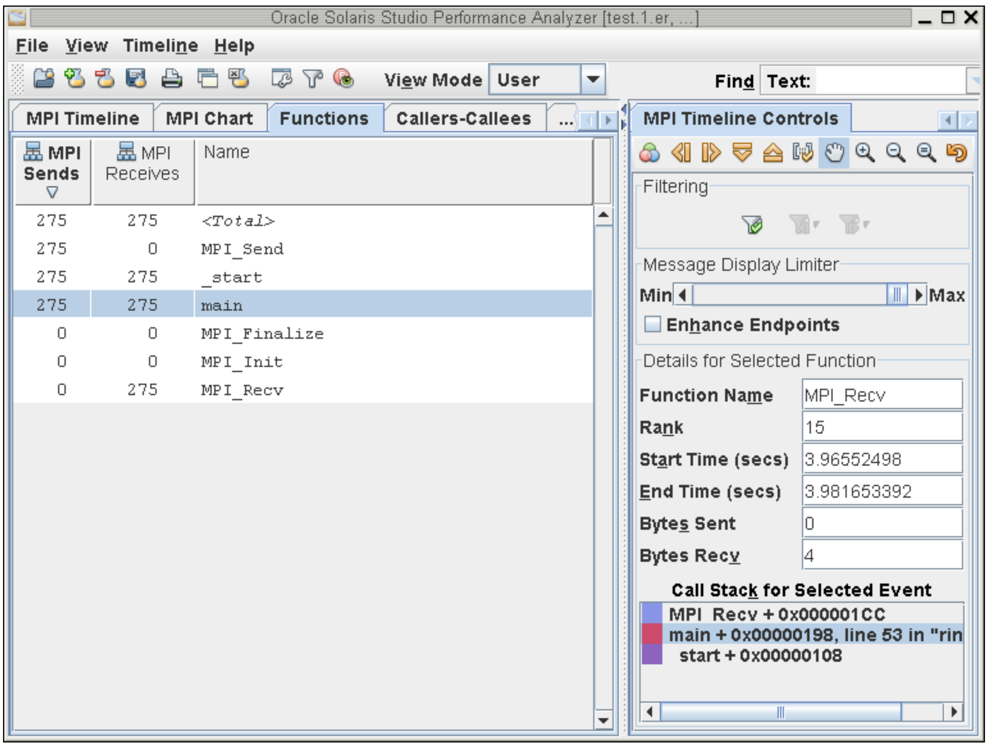


Tip – The screen capture shows a reduced number of metrics. You can select more metrics to display by choosing View > Set Data Presentation > Metrics.

---

5. Click the Functions tab in the main Performance Analyzer panel.

   The Functions tab shows the same MPI Send and MPI Receive metrics in columns on the left side of a table. You can sort the table by clicking in the column headers.



Tip – The screen capture shows a reduced number of metrics. You can select more metrics to display by choosing View > Set Data Presentation > Metrics.

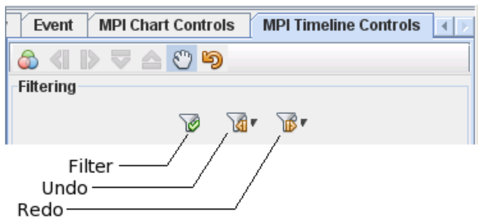
---

6. Click the MPI Timeline tab to return to the MPI timeline.

   Do not click the regular Timeline tab because it does not apply to MPI programs.

Filtering Data in the MPI Tabs

The filtering facility lets you select different views of the collected messaging data. You can undo and redo the filters using the filtering controls in either the MPI Chart Controls tab or the MPI Timeline Controls tab.



The first control filters the data by removing everything that is not currently in view.

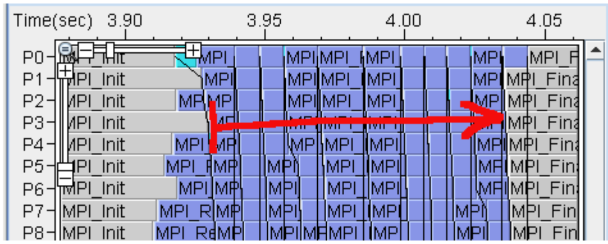
The second control is the Filter Undo button which provides an associated drop down list for removing filters. Clicking this button removes the last filter applied. Clicking the down arrow presents a list of the filters applied, in the order they were applied, with the most recent at the top of the list. When you select a filter in this list, the selected filter and all filters above it on the list are removed.

The third control is the Filter Redo button, and it also has an associated drop down list for reapplying filters. Clicking the button reapplies the last filter that you removed. Clicking the down arrow opens a list of all the filters that have been removed, in the order in which they were removed. When you select a filter in this list, the selected filter and all filters above it on the list are reapplied.

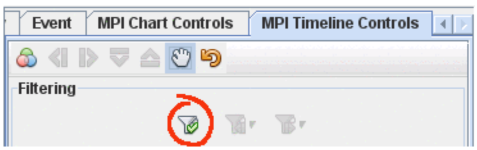
You can redo and undo the filters by using the arrows, similar to going backward and forward in a web browser. You can even remove and apply more than one filter in one click by using the down arrows next to the filter buttons.

The following steps explain how to use a filter to focus on the steady state portion of the program by filtering out the `MPI_Init` and `MPI_Finalize` functions.

1. On the timeline, drag the mouse horizontally to select a region such that `MPI_Init` and `MPI_Finalize` are no longer visible. In the example below, drag from t=3.93 to 4.03.

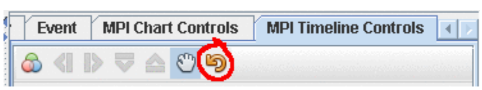


2. Click the Filter button in the MPI Timeline Controls tab.

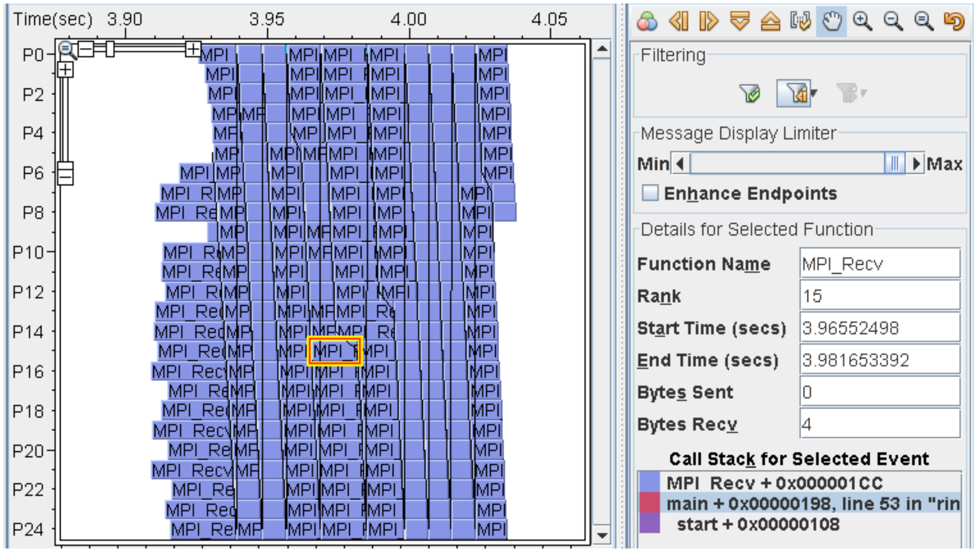


It may look line nothing happened because the filtering is not evident until you change your view by zooming out or by looking at a chart.

3. Click the Zoom Undo button to go back out to the previous zoom.



The display now shows white areas in place of the `MPI_Init` and `MPI_Finalize` functions. White areas indicate that there is no MPI data collected for that area or the data has been filtered out.

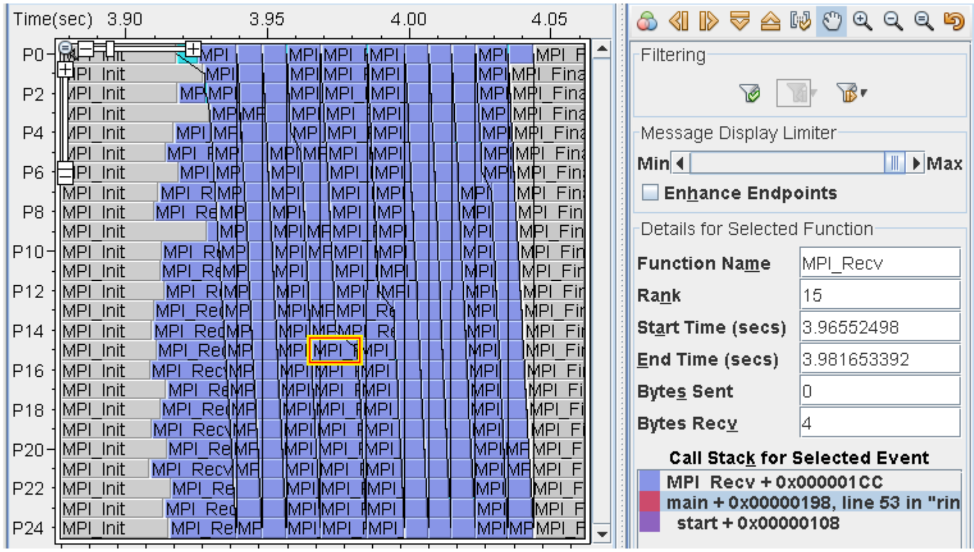


Using the Filter Stack

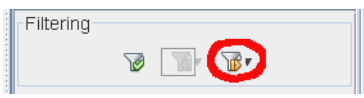
1. Click the Filter Undo drop down arrow to reveal a list of applied filters.

This list lets you choose which filters to remove. It works like a stack: if you select `No filters applied`, everything on top of it will be taken off, which means there will be no filters applied.

2. Select `No filters applied` from the Filter Undo list. The timeline should now look similar to the following.



3. Restore the previous filter using Filter Redo.

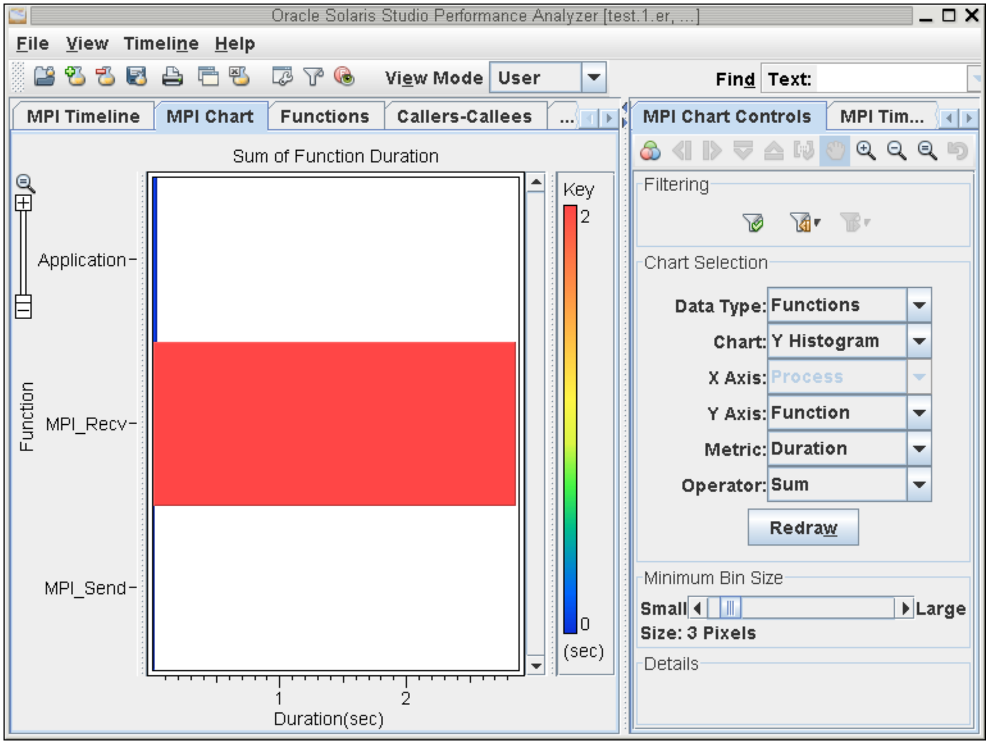


4. Before moving on to the next section, verify that `MPI_Init` and `MPI_Finalize` are filtered out and white areas are displayed as before.

Using the MPI Chart Tab

With `MPI_Init` and `MPI_Finalize` already filtered out, explore the MPI Chart features. The initial chart shows which functions took the most time.

1. Click the MPI Chart tab to see a chart similar to the following.



The MPI Chart tab opens with a chart that shows the sum of the durations of the functions as they ran in all the processes. The vertical rainbow scale to the right of the chart shows the mapping between colors and values. The `MPI_Send` and `Application` functions take almost no time, whereas in the example, the cumulative time in `MPI_Recv` was nearly three seconds.

2. Click on the bar for the `MPI_Recv` function.

   The exact value of the bar is displayed in the MPI Chart Controls tab.

In this particular application, every process waits until the token has passed to every other process. As a result, significant time is spent in `MPI_Recv`, and little time is spent in `Application`, a state that represents time between MPI functions. As you will see later, a delay in message delivery to one rank prevents all other ranks from making progress.

Using the MPI Chart Controls

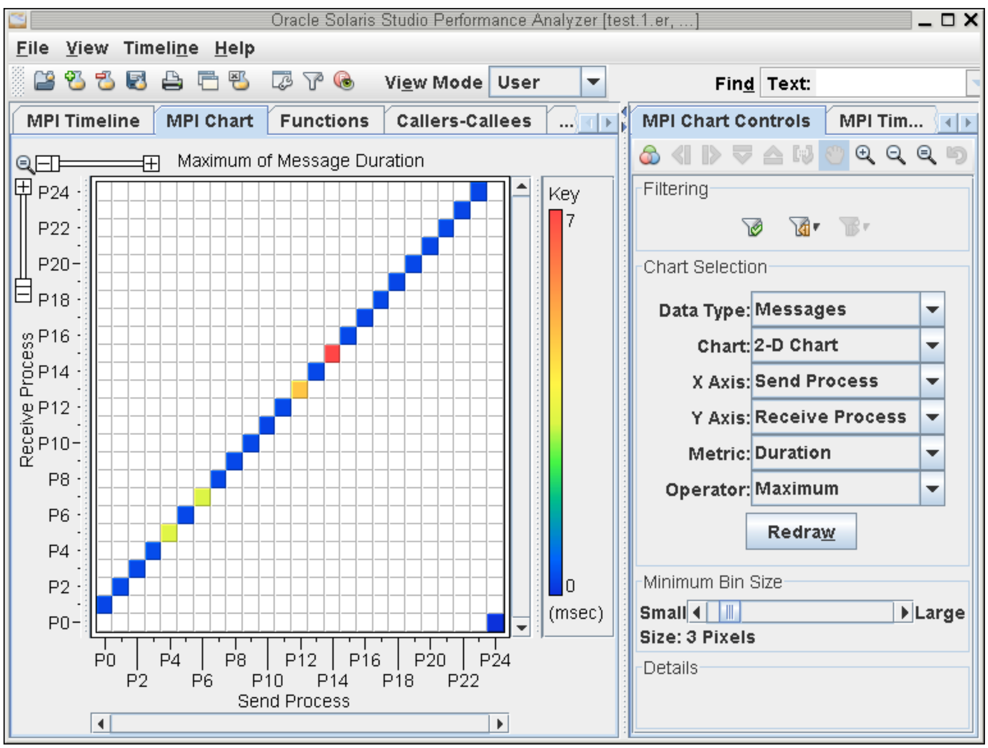
This section shows how to use the MPI Chart Controls tab in different ways to visualize the data. Depending on the program under analysis, some chart options are more useful than others. See Appendix A, "MPI Chart Control Settings," for descriptions of all the MPI Chart controls.

Make a Chart to Show Where Messages are Being Sent

1. Create a chart to look at messages by making the following selections in the MPI Chart Controls tab:

Data Type:	Messages
Chart:	2–D Chart
X Axis:	Send Process
Y Axis:	Receive Process
Metric:	Duration
Operator:	Maximum

2. Click Redraw, and you should see a chart similar to the following:



This chart shows that Process 0 sends only to Process 1. Process 1 only sends to Process 2, and so on. The color of each box is set by the metric selected (Duration) and the operator

(Maximum). Since this graph's Data Type is Messages, this will be the sum of duration of the messages, or the length of message lines in the time dimension.

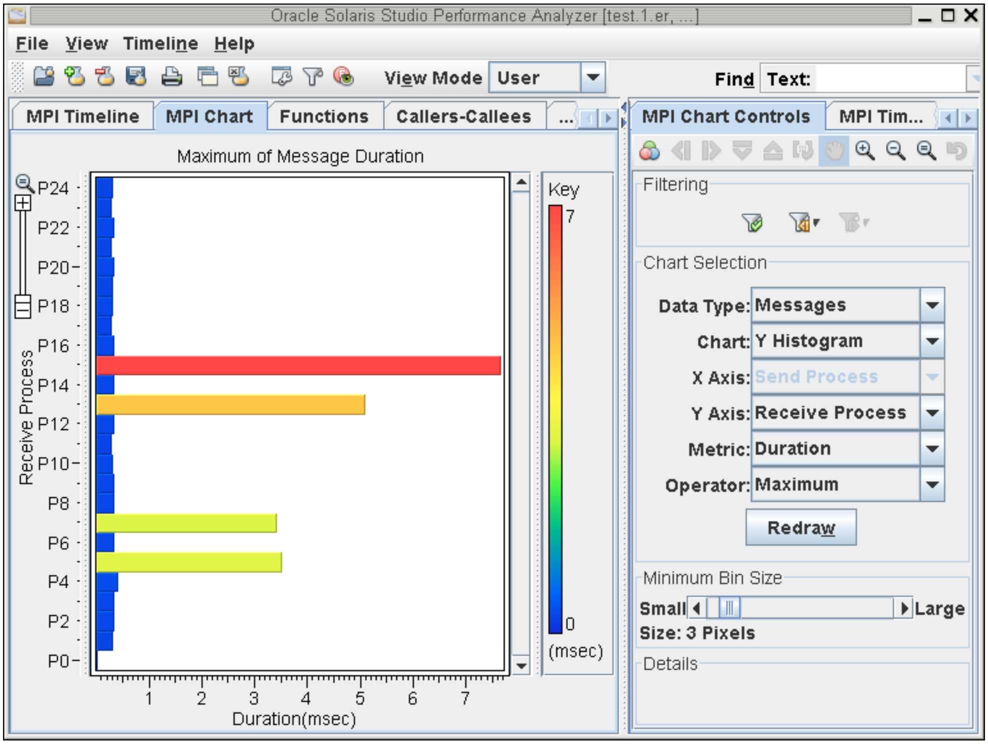
The chart colors indicate maximum message durations between ranks. In this example, the message that took the longest to arrive was sent from P14 to P15.

Make a Chart to Show Which Ranks Waited Longest to Receive a Message

1. Make the following selections in the MPI Chart Controls tab:

Data Type:	Messages
Chart:	Y Histogram
X Axis:	*N/A*
Y Axis:	Receive Process
Metric:	Duration
Operator:	Maximum

2. Click Redraw to draw a new chart

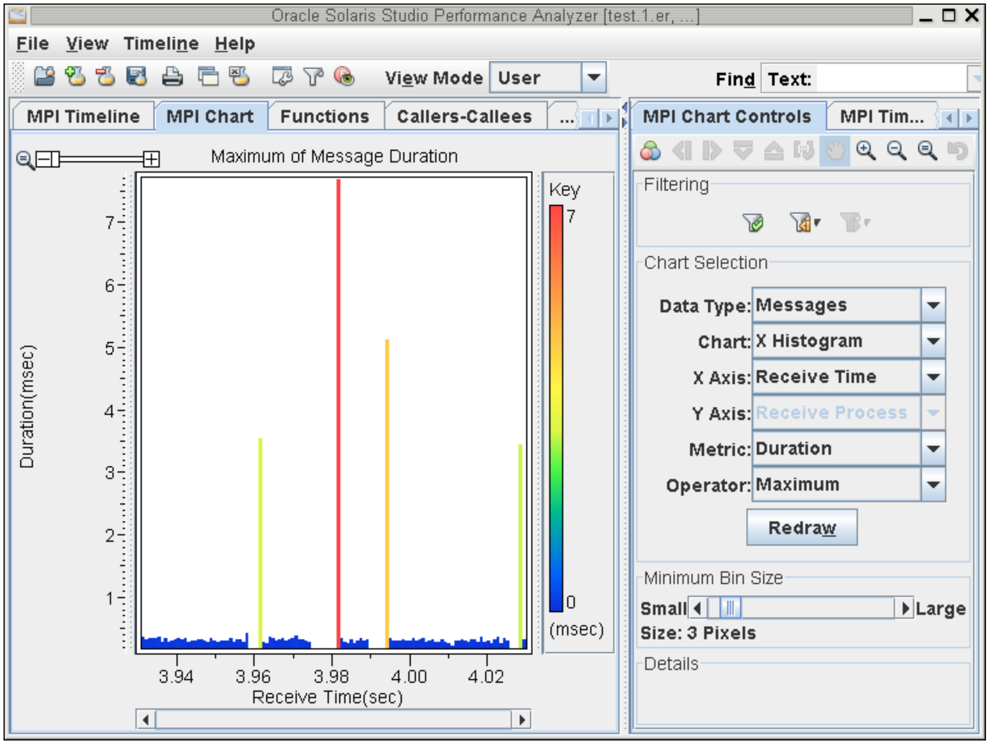


In this example, the chart shows that the P15 rank waited the longest to receive a message.

3. To show a histogram for when these long duration messages occurred, make the following selections for the controls:

Data Type:	Messages
Chart:	X Histogram
X Axis:	Receive Time
Y Axis:	*N/A*
Metric:	Duration
Operator:	Maximum

4. Click Redraw to see a chart similar to the following:



In this example, the slowest message was received at 3.981 seconds.

Look for Slow Message Effects on Time Spent in MPI Functions

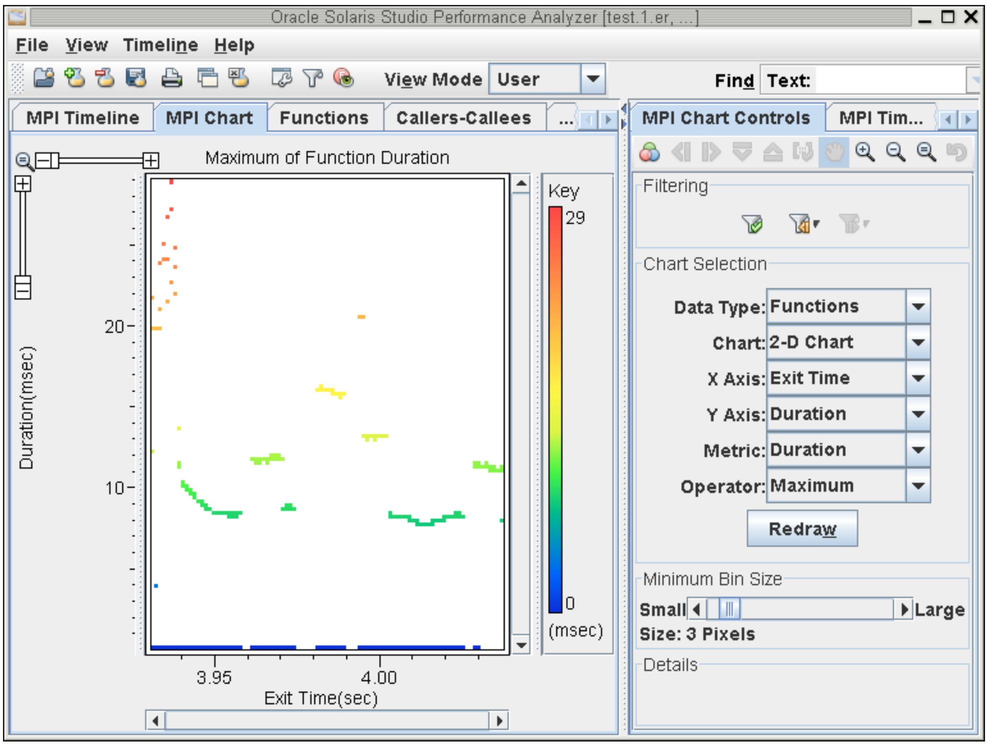
To see the effect of the long duration messages, create a graph that shows duration of functions versus time.

1. Create a chart to look at messages by making the following selections in the MPI Chart Controls tab:

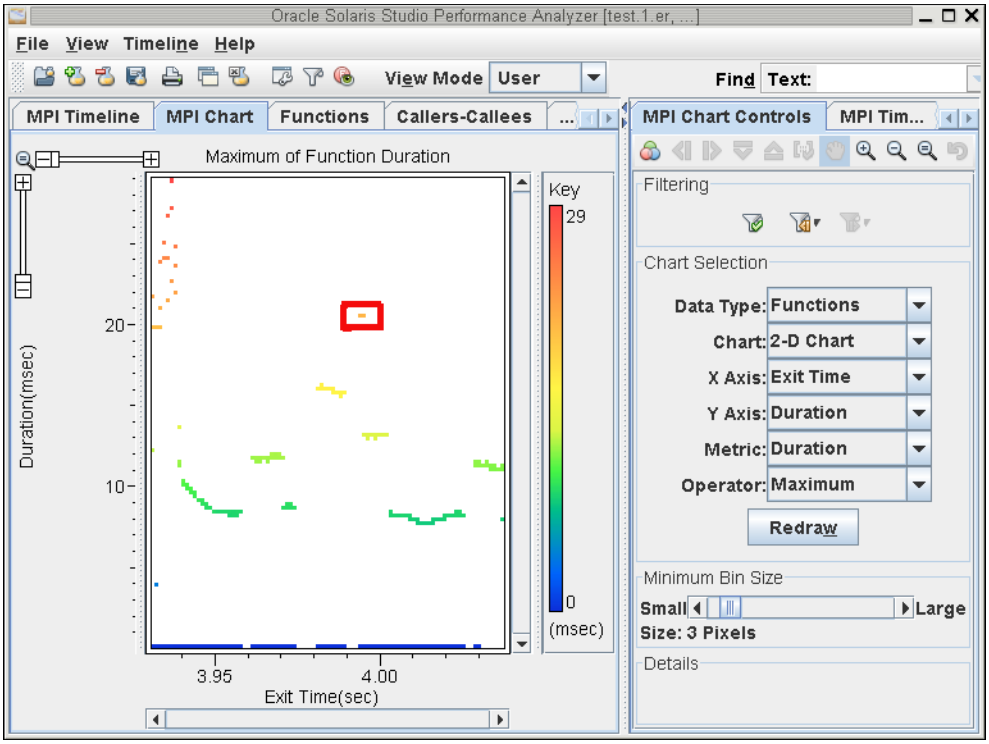
Data Type:	Functions
Chart:	2–D Chart
X Axis:	Exit Time
Y Axis:	Duration
Metric:	Duration
Operator:	Maximum

2. Click Redraw.

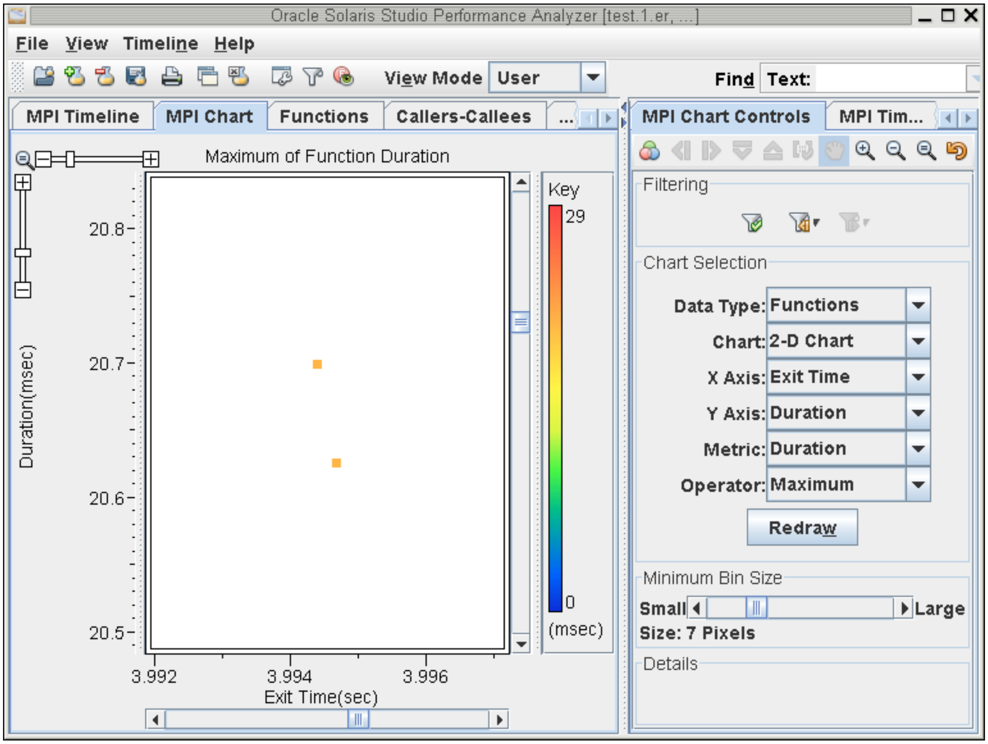
In this example, the graph shows several time regions that have long-duration functions. Note that at t=3.99, there are function durations longer than 20 seconds. This time corresponds to the long message flight-times viewed in the previous chart.



3. Isolate these long duration functions by dragging a box around them to zoom:

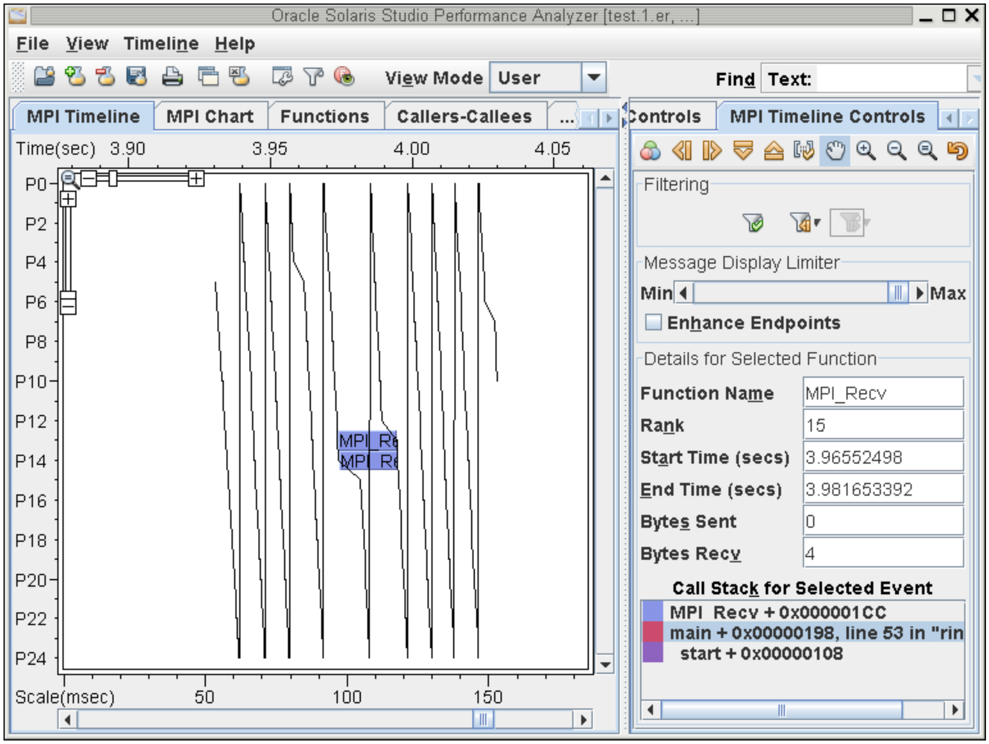


4. Click the Filter button to examine the only these function calls.



5. Click the MPI Timeline tab.

   You can now identify the high duration functions on the MPI Timeline. They are the result of messages with slow delivery times.



6. Click the Filter Undo and Redo buttons to toggle the context around the long duration functions.

For more information about the MPI Chart Controls, see Appendix A, "MPI Chart Control Settings"

Conclusion

Oracle Solaris Studio Performance Analyzer enables you to observe performance and pinpoint problem areas in complex multithreaded applications. The simple example described in this tutorial illustrates the basics for examining relationships between MPI functions and messages. Using the MPI timeline, MPI charts, and zooming and filtering capabilities, you can gather and process performance data, view metrics at the program, function, source line, and instruction level, and identify potential performance problems before they become deployment issues.

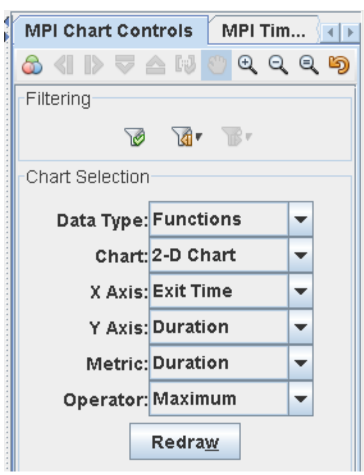
APPENDIX A

MPI Chart Control Settings

This appendix describes all the settings of the MPI Chart Controls tab in the Oracle Solaris Studio Performance Analyzer.

Chart Attributes

This section describes the attributes you can set in the MPI Chart Controls tab to create charts of the MPI experiment data. The MPI Chart Controls tab is shown in the following screen capture.



- Data Type – Controls the type of data that is displayed in the chart, and can be set to one of the following values:

  - Functions - Plot data about the MPI functions used by the program. See "Functions Chart Attributes" on page 36 for attributes you can set for Functions charts.
  - Messages - Plot data about the MPI messages sent between process ranks. See "Messages Chart Attributes" on page 37 for attributes you can set for Messages charts.
- Chart – Controls the type of chart that is created, and can be set to one of the following values:
  - Y Histogram - One dimensional chart with the data plotted on the vertical axis as a function of another metric on the horizontal axis. You must select the type of data to plot on the Y axis and the metric for the X axis.
  - X Histogram - One dimensional chart with the data plotted on the horizontal axis as a function of time on the vertical axis. You must select the type of data to plot on the X axis, and the metric for the Y axis.
  - 2-D chart - Two dimensional chart with data plotted on both X and Y axis, making a 2-D matrix or scatter plot. You must specify what to plot on the X and Y axis, and the metric.
- X Axis - Select the type of data to plot on the horizontal axis, for X Histogram or 2-D Chart. The types available depend on whether you have selected Functions or Messages from the Data Type list. The possible values are described in "Functions Chart Attributes" on page 36 and "Messages Chart Attributes" on page 37.
- Y Axis - Select the type of data to plot on the vertical axis, for Y Histogram or 2-D Chart. The types available depend on whether you have selected Functions or Messages from the Data Type list. The possible values are described in "Functions Chart Attributes" on page 36 and "Messages Chart Attributes" on page 37
- Metric - Select what data is shown as a function of X or Y. The metric value is indicated through color in the charts. The types available depend on whether you have selected Functions or Messages from the Data Type list. The possible values are described in "Functions Chart Attributes" on page 36 and "Messages Chart Attributes" on page 37.
- Operator – Select the method used to combined metrics in the chart. The possible values are described in "Operator Settings" on page 38.

Functions Chart Attributes

The following are the chart attributes you can set when plotting data for Functions. These attributes can be selected for the X Axis, Y Axis, and Metric.

- Time (range) - The range of times from entry to exit of a function
- Entry Time - The time when a function is called
- Exit Time - The time when a function returns to the caller
- Duration - The time difference between function entry and function exit
- Process - The MPI global ranks in numerical order. Each function call has a unique process rank associated with it.

- Function - The MPI function called.
- Send Bytes - Number of bytes sent in an MPI function call
- Receive Bytes - Number of bytes received in an MPI function call
- 1 (only for Metric) - Specifying 1 as the metric simply specifies an attribute whose value is always 1. This can be used to count data records or signal the presence or absence of data. For example, to count the number of function calls for each function, set Y Axis: Function, Metric: 1, Operation: Sum. To detect whether any function calls were made, set Operation: Maximum.

Messages Chart Attributes

The following are the chart attributes you can set when plotting data for Messages. These attributes can be selected for the X Axis, Y Axis, and Metric.

- Time (range) - The range of time from send to receive of a message
- Send Time - The time that a message was sent
- Receive Time - The time that a message was received
- Duration - The time difference between send and receive of a message
- Send Process - The process that sent a message
- Receive Process - The process that received a message
- Communicator - An arbitrarily defined ID that uniquely labels the communicator (set of processes) used to send and receive the message
- Tag - The MPI tag used to identify the message
- Send Function - The function that sent the message
- Receive Function - The function that received the message
- Bytes - Number of bytes in the message
- 1 (only for Metric) - Specifying 1 as the Metric simply specifies an attribute whose value is always 1. This can be used to count data records or signal the presence or absence of data. For example, to count the number of function calls for functions that send a message, set Y Axis to Send Function, set Metric to 1, and set Operator to Sum. To detect whether any function calls were made for each function that sends a message, set Operator to Maximum.

Operator Settings

The Operator control determines how multiple metric values are combined in the chart, and can be set to one of the following values:

- Sum - calculates the sum of the selected Metric, which must be one of: Time, Duration, Send/Receive Bytes, or "1"
- Maximum - calculates the maximum value of the selected Metric, which must be one of: Time, Duration, Send/Receive Bytes, or "1"
- Minimum - calculates the minimum value of the selected Metric, which must be one of: Time, Duration, Send/Receive Bytes, or "1"
- Average - calculates the average value of the selected Metric, which must be one of: Time, Duration, Send/Receive Bytes, or "1"
- Fair - The Fair operator operates on any type of metric. When many metric values are all assigned to the same chart bin, the Fair operator picks a single one of those values "fairly". For example, suppose that 90% of the MPI messages use the communicator `MPI_COMM_WORLD`, but 10% of them use a user-defined communicator `mycomm`. If you create a message chart using "Communicator" as the metric and "Fair" as the operator, the chart would report `MPI_COMM_WORLD` 90% of the time but `mycomm` 10% of the time.