Intel® Trace Analyzer 9.1 Update 2

User and Reference Guide

Contents

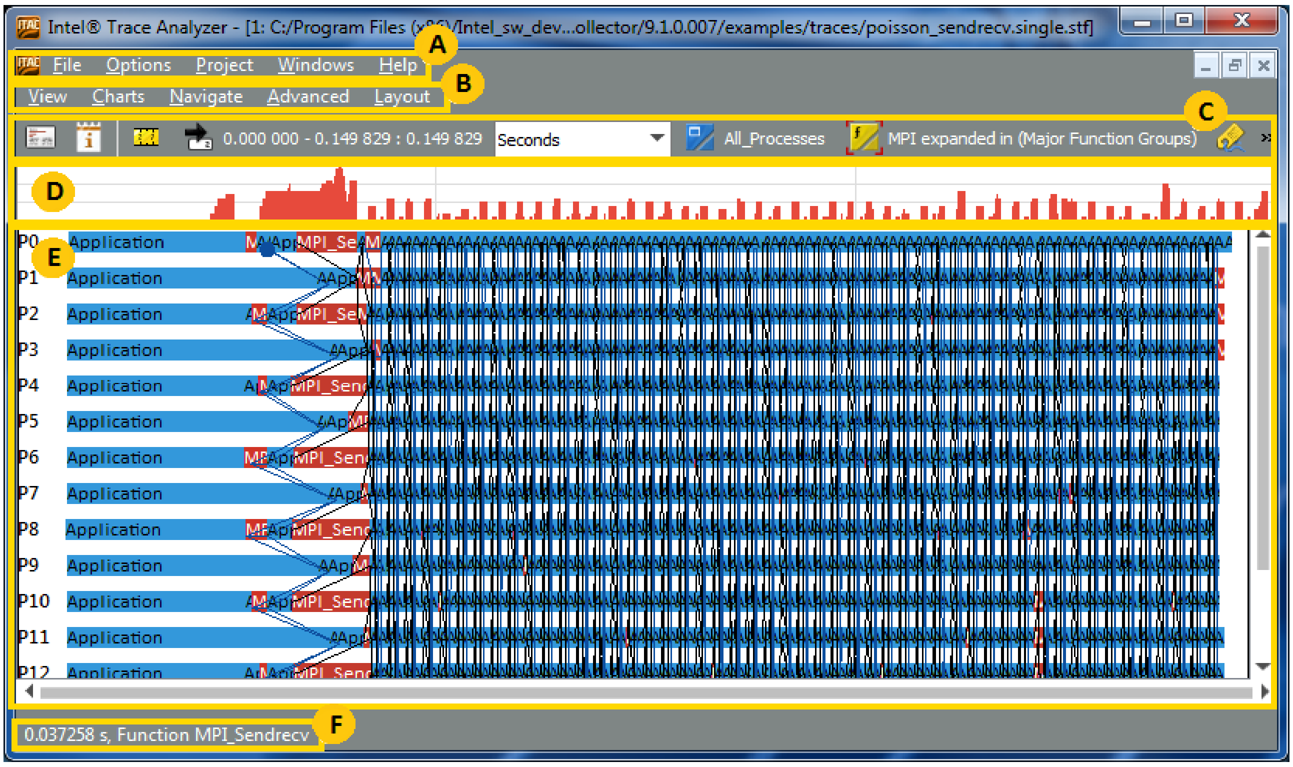
Legal Information

1. Introduction

Intel® Trace Analyzer is a graphical tool that displays and analyzes event trace data generated by Intel® Trace Collector. Intel® Trace Analyzer helps you understand the application behavior, detect performance problems and programming errors.

The tool uses trace files `.stf` as input, which contain the trace data of an MPI application. For instructions on how to generate trace files, see the *Intel® Trace Collector User and Reference Guide*.

Intel® Trace Analyzer GUI


Intel® Trace Analyzer - [1: C:/Program Files (x86)/Intel_sw_dev...ollector/9.1.0.007/examples/traces/poisson_sendrecv.single.stf]
A
File Options Project Windows Help
B
View Charts Navigate Advanced Layout
C
0.000 000 - 0.149 829 : 0.149 829
Seconds
All_Processes
MPI expanded in (Major Function Groups)
D
E
P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12
Application
MPI_Sen
F
0.037258 s, Function MPI_Sendrecv

A	Use the Intel® Trace Analyzer main menu bar to set up your working environment. See Main Menu Bar for details.
B	Use the view menu bar to configure the opened views. See View Menu Bar for details. Use the **Charts** menu to open and navigate the various Intel Trace Analyzer charts within the current trace file and use them to analyze the application trace data.
C	Use the **Toolbar** buttons to configure the currently open trace file view. See Toolbar for details.
D	The **Trace Map** displays the MPI function activity for the application over time. MPI function activity is displayed in red. Use the **Trace Map** to zoom into the relevant subsets of trace file charts. This map appears for all the charts. See Trace Map for details.
E	The currently open chart is the **Event Timeline**. This chart displays individual process activities over time. Horizontal bars represent the processes with the functions called in these processes. Black lines indicate messages sent between processes and blue lines represent collective operations. See Event Timeline for details. To change the displayed chart, select **Charts**. For description of all available charts, see Charts.

F	The **Status Bar** displays the exact time point when you hover the mouse over the processes shown in the **Event Timeline**.

1.1. What's New

This User and Reference Guide documents Intel® Trace Analyzer 9.1 Update 2.

Intel® Trace Analyzer 9.1 Update 2:

- Introduced an interoperability feature with Intel® Advisor XE. See Interoperability with Intel® VTune™ Amplifier XE and Intel® Advisor XE for details.

Intel® Trace Analyzer 9.1:

- The document has been restructured into the User Guide and Reference Manual.
- Configuration Assistant is now accessed through the Intel Trace Analyzer GUI. See the Configuration Assistant section for details.
- Minor improvements and bug-fixes.

Intel® Trace Analyzer 9.0 Update 3:

- Support for OpenMP* regions. See the OpenMP* Regions Support section for details.

Intel® Trace Analyzer 9.0 Update 2:

- Introduced an interoperability feature with Intel® VTune™ Amplifier XE. See Interoperability with Intel® VTune™ Amplifier XE and Intel® Advisor XE for details.

Intel® Trace Analyzer 9.0 Update 1:

The following is a new feature in this version of the Intel Trace Analyzer:

- Updated directory structure. See how to access Intel Trace Analyzer in the Starting Intel® Trace Analyzer section.

Intel® Trace Analyzer 9.0:

- New Performance Assistant chart. See how to use the Performance Assistant in Performance Assistant.
- You can also use the Performance Assistant in the Command Line Interface. See the description of the `--assist` option in the Command Line Interface (CLI) section.
- Added Summary Page. It provides you with general summary on time spent in MPI and gives you hints on how to start the analysis of the application. See the description in the Summary Page section.
- Now you can track I/O calls. To see how they are displayed on the Intel® Trace Collector charts, refer to the Charts section.

Intel® Trace Analyzer 8.1 Update 4:

- This release contains minor improvements and bug-fixes.

Intel® Trace Analyzer 8.1 Update 3:

- Added Trace Map. For details, see Trace Map.
- The Preferences dialog box now contains new tabs for timeline settings. For more information, see Preferences.

Intel® Trace Analyzer 8.1 Update 2:

- Added Toolbar. It provides quick access to the Intel Trace Analyzer functionality. For details, see Toolbar.
- Added Event Timeline preferences tab to the **Preferences** dialog box. For more information, see Preferences.

Intel® Trace Analyzer 8.1 Update 1:

- Added Welcome Page. It provides a quick start to operating Intel Trace Analyzer. For details, see Welcome Page.
- Added New Preferences dialog. It enables you to set up your working environment. See Preferences.

1.2. About this Document

This *User and Reference Guide* provides you with the description of the features of the Intel® Trace Collector. This information is provided in the two main sections:

- User Guide – provides the usage instructions on Intel® Trace Analyzer and description of all its GUI elements.
- Intel® Trace Analyzer Reference – provides the reference information for Intel® Trace Analyzer and introduces its main concepts.

1.2.1. Notational Conventions

The following conventions are used in this document.

Convention	**Explanation**	**Example**
This type style	Introduces new terms, denotation of terms, placeholders, or titles of manuals.	The term *process* in this documentation implicitly includes thread.
This type style	Denotes GUI elements	Click **OK**
This type style	Hyperlinks	https://premier.intel.com/
`This type style`	Commands, arguments, options	`traceanalyzer --cli trace.stf -c0 -w`
`<`*`this type style`*`>`	Placeholders for actual values	`<`*`new_name`*`>`
`$`	Introduces UNIX* commands	`$ ls`
>	Indicates a menu item inside a menu	**Main Menu > File > Quit**

NOTE

The term *process* in this documentation implicitly includes thread. As soon as the Intel® Trace Analyzer loads a trace file that was generated running a multithreaded application, the GUI uses the term *thread* only if it is applicable. This is done to avoid confusing MPI application programmers who normally use the term process instead of thread.

1.3. Related Information

Additional information about Intel® Trace Analyzer for Linux* OS, Windows* OS and OS X* related products is available at: http://software.intel.com/en-us/articles/intel-trace-analyzer-and-collector-documentation/

Intel® Premier Customer Support is available at: https://premier.intel.com/

Submit your feedback on the documentation at:
http://www.intel.com/software/products/softwaredocs_feedback/

2. User Guide

2.1. Starting Intel® Trace Analyzer

2.1.1. Starting on Linux* OS

Invoke Intel® Trace Analyzer through the command line:

```
$ traceanalyzer
```

Optionally, specify one or more trace files as arguments. For example:

```
$ traceanalyzer poisson_icomm.single.stf
```

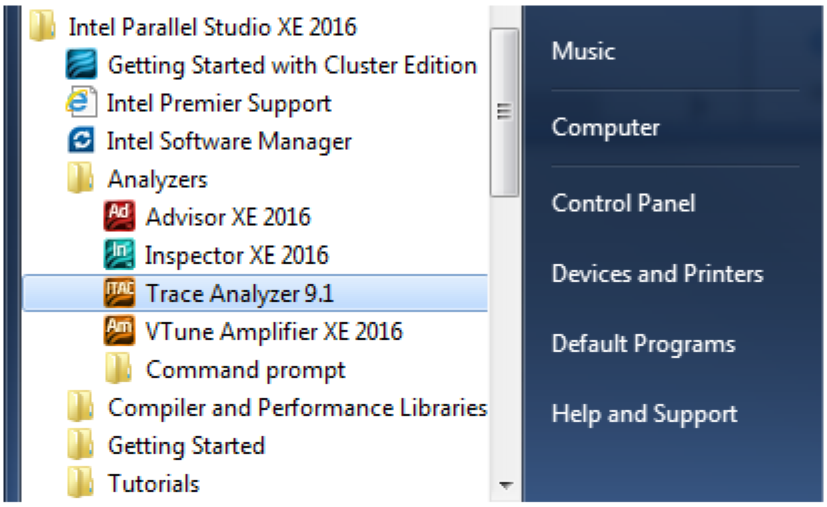
The example above opens the trace file `poisson_icomm.single.stf` in Intel Trace Analyzer.

To open trace files in the Intel Trace Analyzer without restarting it, select **File > Open**.

2.1.2. Starting on Windows* OS

Use one of the following options to invoke the Intel Trace Analyzer:

- Double-click on a trace file.
- Go to **Start > All Programs > Intel Parallel Studio XE *version* > Analyzers > Intel Trace Analyzer *version***.



- Run the command:

  ```
  > traceanalyzer
  ```

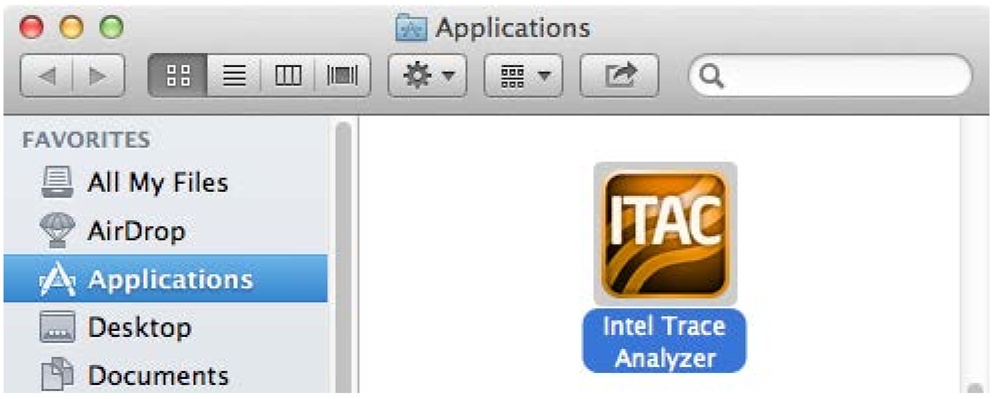
  Optionally, specify one or more trace files as arguments.

To open trace files in the Intel Trace Analyzer without restarting it, select **File > Open**.

2.1.3. Starting on OS X*

Use one of the following options to invoke the Intel Trace Analyzer:

- On the menu bar select **Go > Applications > Intel Trace Analyzer**.



- Run the command:

```
$ traceanalyzer
```

  Optionally, specify one or more trace files as arguments.

To open trace files in the Intel Trace Analyzer without restarting it, select **File > Open**.

NOTE

For proper functioning of Intel Trace Analyzer, ensure that files do not get modified while they are opened in Intel Trace Analyzer.

2.1.4. Command Line Interface

Intel Trace Analyzer provides a command line interface (CLI) in all environments.

Use the CLI to:

- Pre-calculate trace caches for new trace files from batch files without invoking the graphical user interface. Use it for very big trace files.
- Automate profiling data production for several trace files without further interaction.

For every opened trace file, Intel Trace Analyzer creates a trace cache. It is stored in the same directory as the trace file and has the same file name adding the `.cache` suffix. If Intel Trace Analyzer cannot create a cache file in the default directory, it can create it in the system temporary directory.

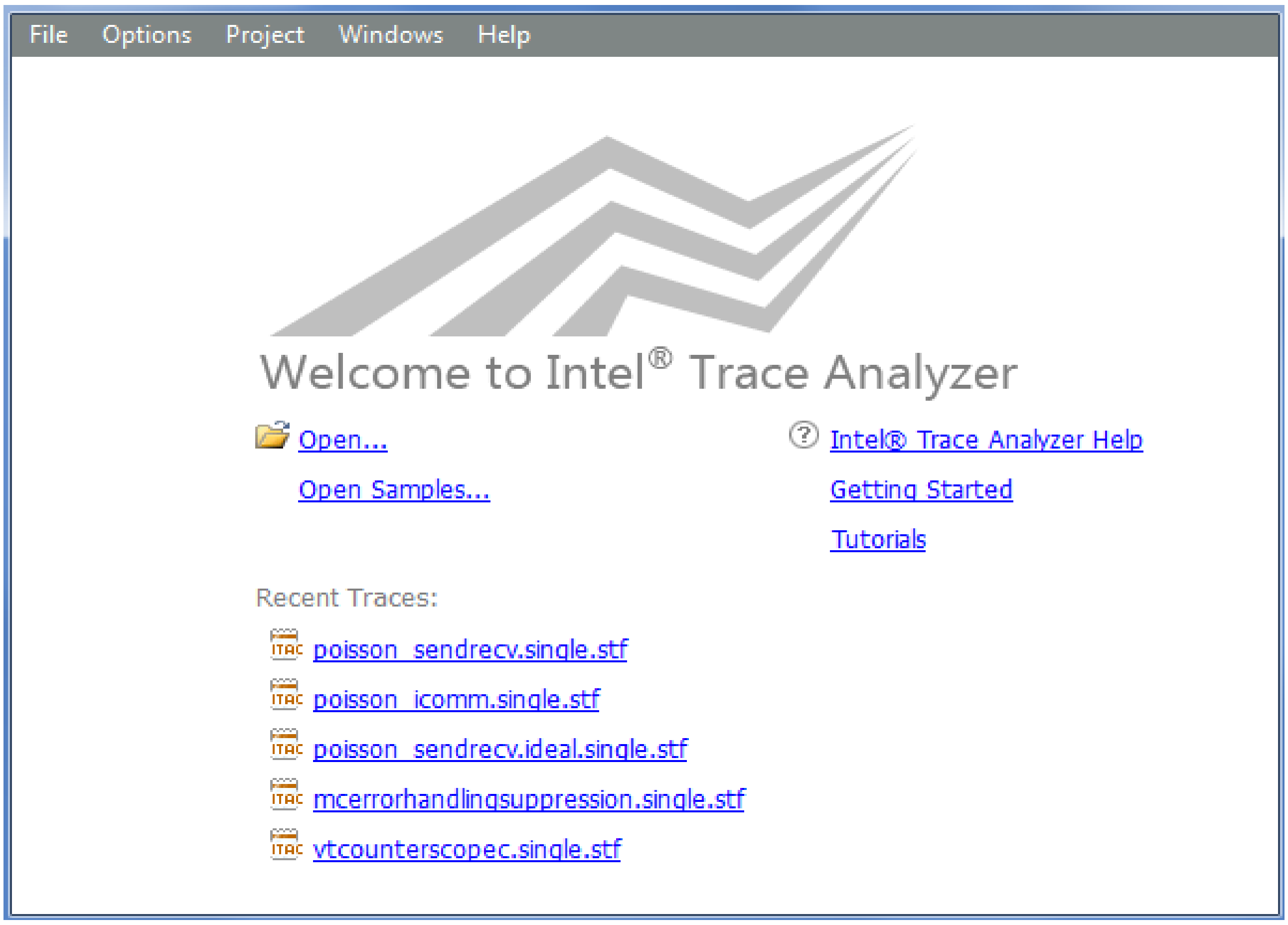
See Also

Command Line Interface (CLI)

2.2. Graphical User Interface

2.2.1. Welcome Page

Welcome Page provides a quick start to operating the Intel® Trace Analyzer.



Welcome to Intel® Trace Analyzer

Open...

Open Samples...

Intel® Trace Analyzer Help

Getting Started

Tutorials

Recent Traces:

poisson_sendrecv.single.stf

poisson_icomm.single.stf

poisson_sendrecv.ideal.single.stf

mcerrorhandlingsuppression.single.stf

vtcounterscopec.single.stf

From the Welcome Page you can:

Select:	**To Open:**
Open...	Tracefile/Project dialog box
Open Samples...	Intel Trace Analyzer sample tracefiles
Intel® Trace Analyzer Help	Intel® Trace Analyzer User and Reference Guide
Getting Started	Getting Started page that describes the essential steps to start working with the Intel® Trace Analyzer and Collector
Tutorials	Index file listing the Intel® Trace Analyzer and Collector tutorials
Recent Traces	Recently used tracefiles

You can also find **Open...** menu and recently opened tracefiles in the File menu and **Intel® Trace Analyzer Help** in the Help menu.

See Also

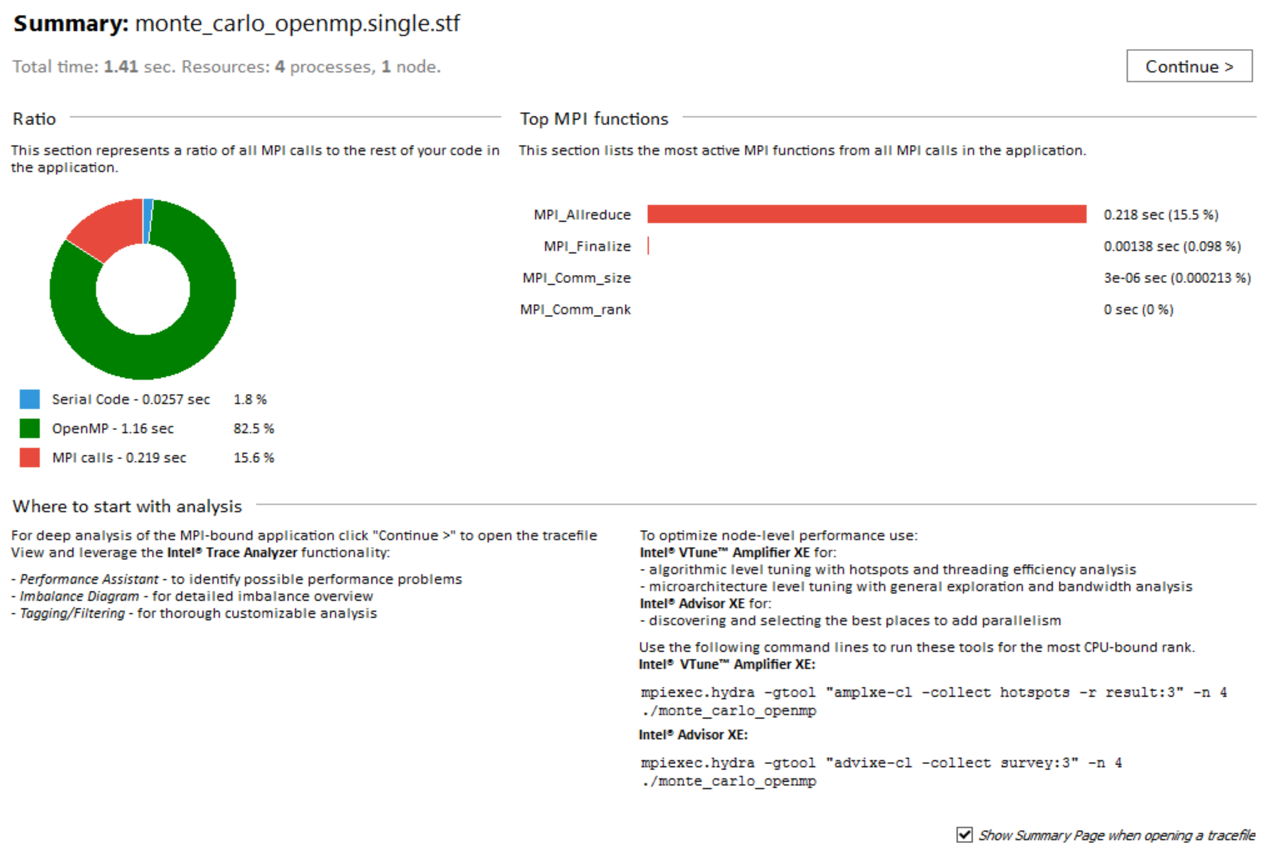
File Menu
Help Menu

2.2.2. Summary Page

Summary page provides you with summary information on the time your application spent in MPI.

To access the Summary page, go to **View > Summary Page**, or click the toolbar button.

If you do not want the Summary page to appear each time you open a new trace file in the Intel® Trace Analyzer, uncheck the **Show Summary Page when opening a tracefile** check box on the Summary Page or in **Options > Preferences > Tracefile Preferences**.


Summary: monte_carlo_openmp.single.stf

Total time: **1.41** sec. Resources: **4** processes, **1** node.

This section represents a ratio of all MPI calls to the rest of your code in the application.

🟦	Serial Code - 0.0257 sec	1.8 %
🟩	OpenMP - 1.16 sec	82.5 %
🟥	MPI calls - 0.219 sec	15.6 %

This section lists the most active MPI functions from all MPI calls in the application.

MPI_Allreduce	0.218 sec (15.5 %)
MPI_Finalize	0.00138 sec (0.098 %)
MPI_Comm_size	3e-06 sec (0.000213 %)
MPI_Comm_rank	0 sec (0 %)

Where to start with analysis

For deep analysis of the MPI-bound application click "Continue >" to open the tracefile View and leverage the **Intel® Trace Analyzer** functionality:

- *Performance Assistant* - to identify possible performance problems
- *Imbalance Diagram* - for detailed imbalance overview
- *Tagging/Filtering* - for thorough customizable analysis

To optimize node-level performance use:
Intel® VTune™ Amplifier XE for:
- algorithmic level tuning with hotspots and threading efficiency analysis
- microarchitecture level tuning with general exploration and bandwidth analysis
Intel® Advisor XE for:
- discovering and selecting the best places to add parallelism

Use the following command lines to run these tools for the most CPU-bound rank.
Intel® VTune™ Amplifier XE:

```
mpiexec.hydra -gtool "amplxe-cl -collect hotspots -r result:3" -n 4
./monte_carlo_openmp
```

Intel® Advisor XE:

```
mpiexec.hydra -gtool "advixe-cl -collect survey:3" -n 4
./monte_carlo_openmp
```

[✓] *Show Summary Page when opening a tracefile*

Use the Summary page to get the following information:

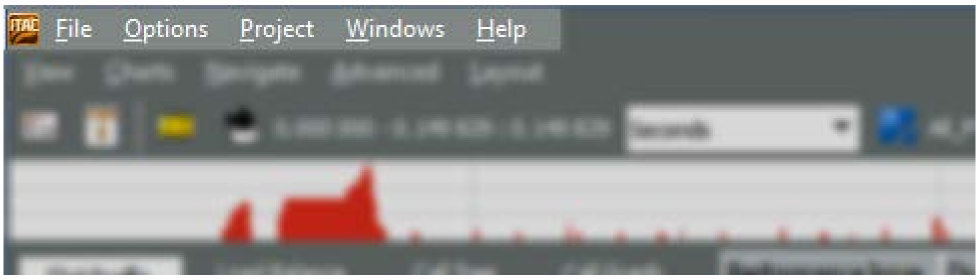
Section:	**Description:**
Continue	Open the standard Intel® Trace Analyzer View
Ratio	Get the ratio of MPI calls and OpenMP* regions to the rest of your code in the application
Top MPI functions	See the most active MPI functions from total MPI in your application
Where to start with analysis	Find advice on where to start analysis of communications in the MPI-bound application and where to optimize the application's performance on the node level Get a command line for analyzing the most CPU-bound process with Intel® VTune™ Amplifier XE or Intel® Advisor XE
Show Summary Page when opening a tracefile	Get the Summary Page for each new tracefile that you open

See Also

Welcome Page
Main Menu
Views

2.2.3. Main Menu Bar

Use the Intel® Trace Analyzer main menu bar to work with files and projects, change the configuration settings and the display style and get help.

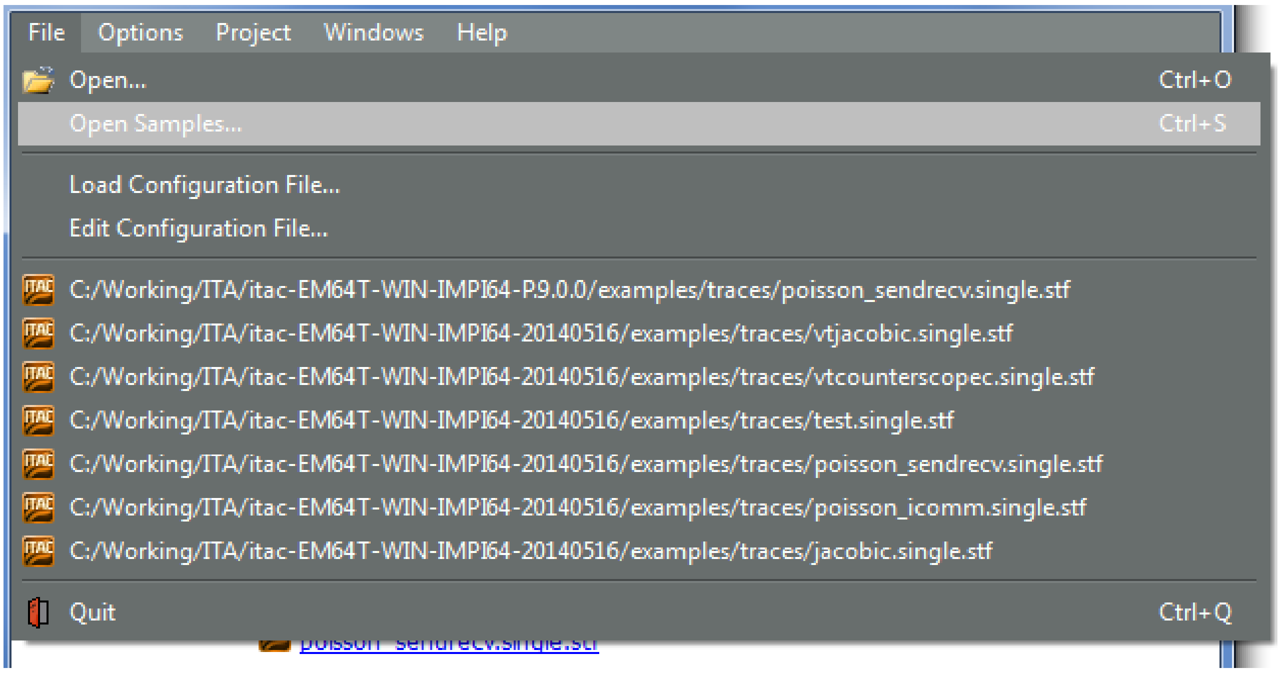


See description of the menus:

- File Menu
- Options Menu
- Project Menu
- Windows Menu
- Help Menu

File Menu

The **File** menu has the following entries: **Open**, **Open Samples**, **Load Configuration**, **Edit Configuration** and **Quit**.


File
Options
Project
Windows
Help
Open... Ctrl+O
Open Samples... Ctrl+S
Load Configuration File...
Edit Configuration File...
C:/Working/ITA/itac-EM64T-WIN-IMPI64-P.9.0.0/examples/traces/poisson_sendrecv.single.stf
C:/Working/ITA/itac-EM64T-WIN-IMPI64-20140516/examples/traces/vtjacobic.single.stf
C:/Working/ITA/itac-EM64T-WIN-IMPI64-20140516/examples/traces/vtcounterscopec.single.stf
C:/Working/ITA/itac-EM64T-WIN-IMPI64-20140516/examples/traces/test.single.stf
C:/Working/ITA/itac-EM64T-WIN-IMPI64-20140516/examples/traces/poisson_sendrecv.single.stf
C:/Working/ITA/itac-EM64T-WIN-IMPI64-20140516/examples/traces/poisson_icomm.single.stf
C:/Working/ITA/itac-EM64T-WIN-IMPI64-20140516/examples/traces/jacobic.single.stf
Quit Ctrl+Q

Select:	Keyboard Shortcut:	To Do:
Open...	`Ctrl+O`	Open one or more trace files
Open Samples...	`Ctrl+S`	Open sample trace files

Load Configuration	none	Select a new configuration
Edit Configuration	none	Edit configuration, for example, by changing values or removing entries from the configuration.
Quit	`Ctrl+Q`	Exit the Intel® Trace Analyzer

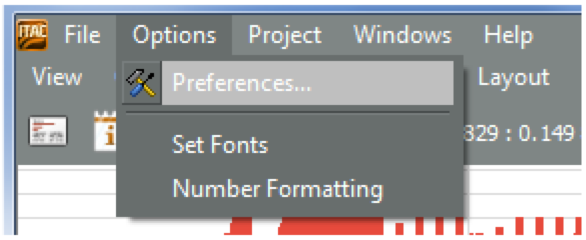
A list of the ten most recently opened tracefiles is appended to the menu.

See Also

Edit Configuration Dialog

Options Menu

The **Options Menu** has the following entries: **Preferences**, **Set Fonts** and **Number Formatting**.



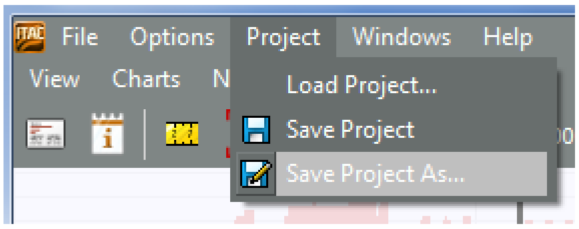
Select:	**To Do:**
Preferences	Set up your working environment
Set Fonts	Change the fonts of the individual Charts
Number Formatting	Customize the format of the various numerical values displayed in the Intel Trace Analyzer

See Also

Preferences
Font Settings
Number Formatting Settings

Project Menu

The **Project** menu enables you to save your working environment. The Intel® Trace Analyzer and Collector does not save your environment automatically. You need to save the environment manually.



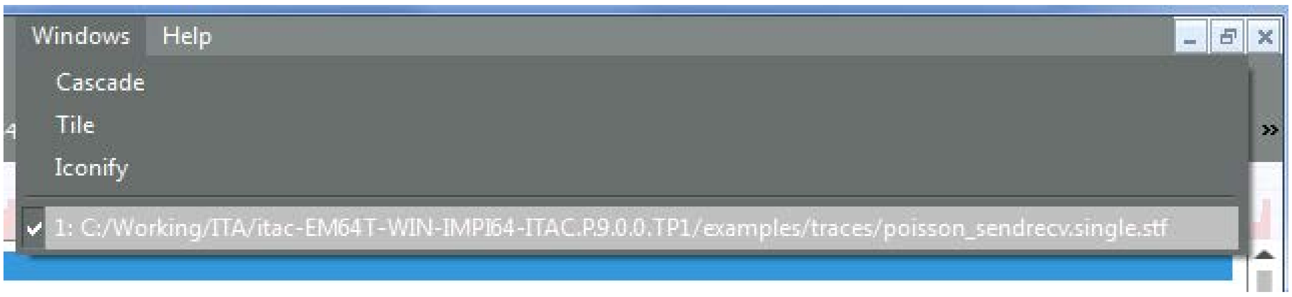
You can perform the following actions from the Project menu:

Setting	**Description**
Load Project...	Open a previously saved project file

Save Project	Save a project using the default naming
Save Project As...	Save a project to a specific folder or project name

Windows Menu

Use the menu options in the **Windows** menu to arrange the open sub-windows as desired. The **Windows** sub-menu also shows the name and path of the trace file that is presently opened.



1: C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9.0.0.TP1/examples/traces/poisson_sendrecv.single.stf

There are three possibilities to use the **Windows** menu:

Setting	Description
Cascade	Arrange the open sub-windows one behind the other with visible title bars
Tile	Arrange the sub-windows next to each other
Iconify	Minimize the open sub-windows and show their icons in the bottom of the View

NOTE

The **Tile** option is useful only when charts are opened in two or more sub-windows.

Help Menu

The **Help** option enables you to access the HTML formatted help through the integrated HTML browser.

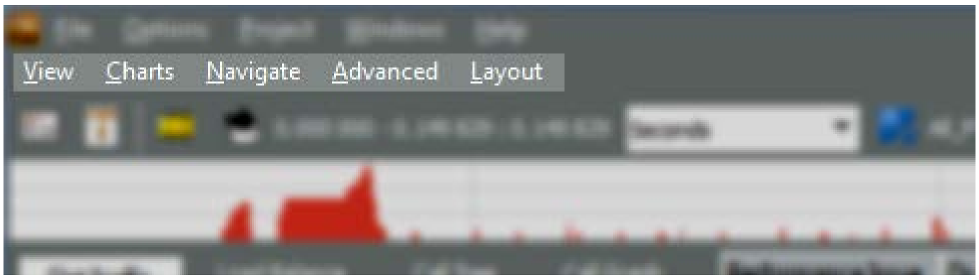
Select:	To Open:
Intel® Trace Analyzer Help	Intel® Trace Analyzer User and Reference Guide
Getting Started	Getting Started page that describes the initial steps to start using the Intel® Trace Analyzer and Collector
Tutorials	Index file listing the Intel® Trace Analyzer and Collector tutorials
Intel® Trace Analyzer and Collector Online	Intel® Trace Analyzer and Collector support web page
Intel® Premier Customer Support	Intel® Premier Customer Support web site where you can submit technical support issues and monitor the previously submitted issues

About Intel® Trace Analyzer	Version and copyright information for the Intel® Trace Analyzer in use

To ease printing, the *Intel® Trace Analyzer User and Reference Guide* is also provided in the PDF format.

2.2.4. View Menu Bar

Use the View menu bar to configure the opened Views.

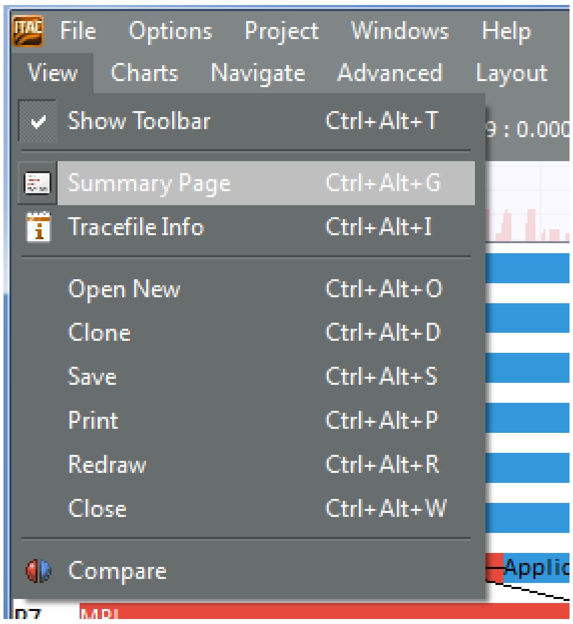


See description of the menus:

- View
- Charts
- Navigate
- Advanced
- Layout
- Comparison

View

The **View** contains the options that apply to the entire View.



These options are:

Option:	Keyboard Shortcut:	Description:
Show Toolbar	`Ctrl+Alt+T`	Have easy access to the Intel® Trace Analyzer functionality that is most often used

Summary Page	`Ctrl+Alt+G`	Get summary information about the time your application spent in MPI
Tracefile Info	`Ctrl+Alt+I`	Get information about the current trace
Open New	`Ctrl+Alt+O`	Open a new View with the Charts previously selected in the **Tracefile preferences** tab of the Preferences dialog box.
Clone	`Ctrl+Alt+D`	Create a one-to-one clone of the current View
Save	`Ctrl+Alt+S`	Save the entire View as a picture
Print	`Ctrl+Alt+P`	Print a copy of the View
Redraw	`Ctrl+Alt+R`	Repaint the entire View
Close	`Ctrl+Alt+W`	Close the current View. If the last View for a trace file is closed, the trace file is closed as well.
Compare	None	Choose the trace files for comparison and compare them with the help of the Comparison dialog box. The traces may represent two runs or two ranges of the same run. Comparison dialog box displays data from the two trace files.
Configure Trace Collector	None	Open the Configuration Assistant

See Also

Preferences
Comparing Two Trace Files
Summary Page

Charts

Select the Charts to open various Intel® Trace Analyzer Charts for use during trace analysis.

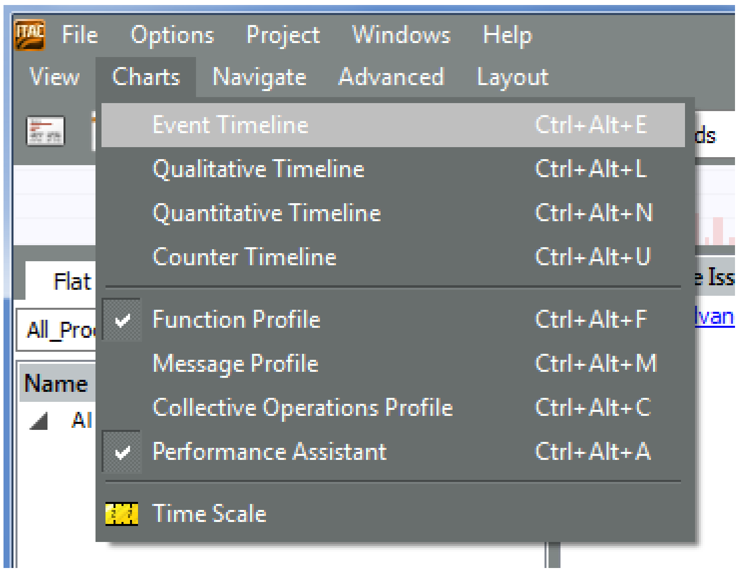

File
Options
Project
Windows
Help
View
Charts
Navigate
Advanced
Layout
Event Timeline Ctrl+Alt+E
Qualitative Timeline Ctrl+Alt+L
Quantitative Timeline Ctrl+Alt+N
Counter Timeline Ctrl+Alt+U
Function Profile Ctrl+Alt+F
Message Profile Ctrl+Alt+M
Collective Operations Profile Ctrl+Alt+C
Performance Assistant Ctrl+Alt+A
Time Scale

Chart:	**Keyboard Shortcut:**	**Description:**
Event Timeline	`Ctrl+Alt+E`	Displays the individual process activity.
Qualitative Timeline	`Ctrl+Alt+L`	Displays event attributes as they occur over time.
Quantitative Timeline	`Ctrl+Alt+N`	Displays the parallel behavior of the application.
Counter Timeline	`Ctrl+Alt+U`	Displays the values of all counters that are present in the trace file.
Function Profile	`Ctrl+Alt+F`	Displays the Function Profile. By default, the Function Profile opens for all newly opened trace files.
Message Profile	`Ctrl+Alt+M`	Displays the statistical information on MPI point to point messages.
Collective Operations Profile	`Ctrl+Alt+C`	Displays the statistical information on MPI collective operations.
Performance Assistant	`Ctrl+Alt+A`	Provides you with the general and detailed information about performance problems.
Time Scale	None	Displays the time scale of the project.

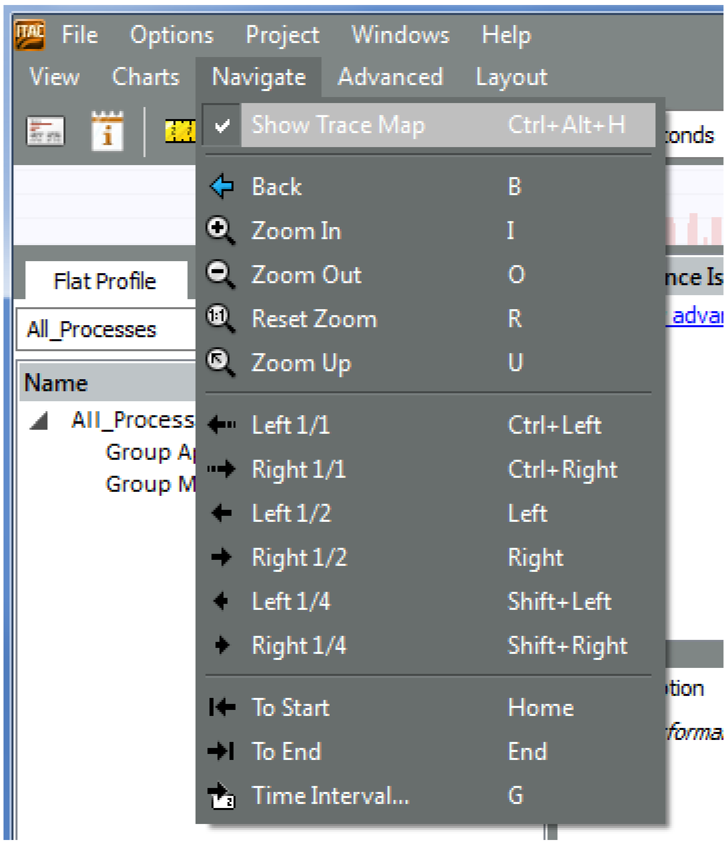
See Also

Event Timeline
Qualitative Timeline
Quantitative Timeline
Counter Timeline
Function Profile Chart

Message Profile
Collective Operations Profile

Navigate

All navigation options are also available through keyboard shortcuts. These keyboard shortcuts work only when a timeline has the keyboard focus. Left-click on a timeline to give it keyboard focus. A black rectangle around the timeline indicates it has the keyboard focus.



Navigation:	**Keyboard Shortcut:**	**Description:**
Show Trace Map	`Ctrl+Alt+H`	Enable/disable the Trace Map
Back	B	Display the current interval from the zoom stack.
Zoom In	I	Scale in by a factor of 0.5 around the current center and push this new interval onto the zoom stack.
Zoom Out	O	Scale out by a factor of 0.5 around the current center and push this new interval onto the zoom stack.
Reset Zoom	R	Clear the zoom stack and push a time interval corresponding to the complete trace file onto the zoom stack.
Zoom Up	U	Maintain the current center and go back to the previous zoom step.

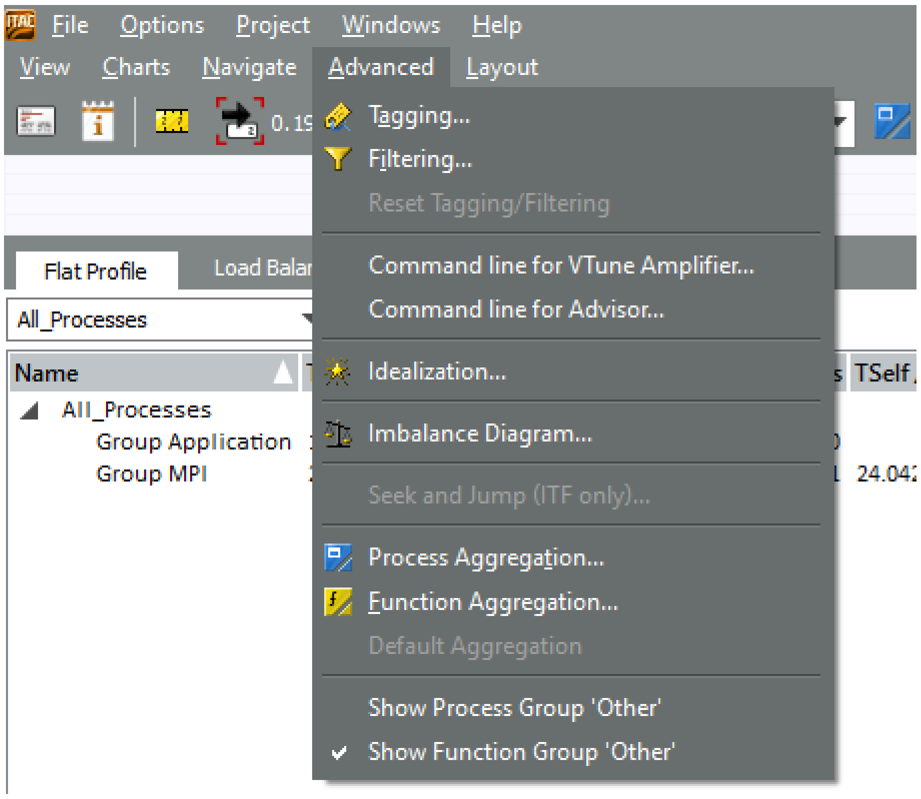
Left 1/1	`Ctrl+Left`	Move one whole screen to the left and push the new interval onto the zoom stack.
Right 1/1	`Ctrl+Right`	Move one whole screen to the right and push the new interval onto the zoom stack.
Left 1/2	`Left`	Move half a screen to the left and push the new interval onto the zoom stack.
Right 1/2	`Right`	Move half a screen to the right and push the new interval onto the zoom stack.
Left 1/4	`Shift+Left`	Move a quarter of a screen to the left and push the new interval onto the zoom stack.
Right 1/4	`Shift+Right`	Move a quarter of a screen to the right and push the new interval onto the zoom stack.
To Start	`Home`	Move to the start of the trace file and push the new interval onto the zoom stack.
To End	`End`	Move to the end of the trace file and push the new interval onto the zoom stack.
Time Interval	`G`	Open the **Time Interval Selection** dialog box.

See Also

Trace Map
Zoom Stack
Time Interval Selection

Advanced

Select Advanced to tag, filter, create an idealize of your trace files, check for imbalance in your application, aggregate processes and functions, go to a particular function event in the trace, or create a command line for Intel® VTune™ Amplifier XE/Intel® Advisor XE.



See the description of the following Advanced menu options:

- Tagging Specific Events
- Filtering Visible Events
- Simulating Ideal Communication
- Checking Application Imbalance
- Aggregating Results
- Aggregating Functions
- Creating a command line for Intel® VTune™ Amplifier XE and Intel® Advisor XE

Tagging Specific Events

You can tag, or highlight events that satisfy specific conditions.

Go to **Advanced** > **Tagging...** to set the conditions for highlighting.

To remove previously set tags and filters, select **Advanced** > **Reset Tagging/Filtering**. This command removes all tags and filters.

See Also

Tagging Dialog Box

Filtering Events

You can filter which events are displayed in the views. If you use filtering, you see only those events that satisfy specific condition(s). All other events are suppressed as if they were not written to the trace file.

To define the filter conditions, select **Advanced** > **Filtering**.

To remove all defined filters and tags, select **Advanced** > **Reset Tagging/Filtering**. This command removes all tags and filters.

See Also

Filtering Dialog Box

Simulating Ideal Communication

Use the Idealization capability to understand application imbalance and estimate a potential application speedup after MPI implementation tuning and/or network upgrades.

The idealizer creates the ideal communication environment from the real trace generated by the Intel® Trace Collector. To simulate the application behavior in the ideal communication environment, select **Advanced** > **Idealization**.

Idealization assumes the following conditions at trace processing time:

- zero network latency
- infinite network bandwidth
- zero buffer copy time
- infinite buffer size

Idealization assumes concurrency as a limitation factor. For example:

- a message cannot be received before it is sent
- an All-To-All collective operation ends when the last thread starts

See Also

Idealization Dialog Box

Checking Application Imbalance

Use the Imbalance Diagram to compare the performance between a real trace and an ideal trace.

To do this:

1. Create an idealized trace file in **Advanced > Idealization**
2. Go to **Advanced > Imbalance Diagram**
3. Identify the traces to compare in the dialog box that appears
4. Select the display mode in the resulting **Imbalance Diagram** dialog box

See Also

Imbalance Diagram Dialog Box

Aggregating Results

Use the Process Aggregation capability to set the process groups to focus on and aggregate the results.

To start, go to **Advanced** > **Process Aggregation**.

You can find the processes that are present in the trace file, but not in the process group, in the **Other** group. By default, the data for the **Other** group is not displayed. To view data for the **Other** group, select **Advanced** > **Show Process Group 'Other'**.

To revert to the default aggregation of all processes, select **Advanced > Default Aggregation**.

See Also

Process Aggregation

Aggregating Functions

Use the Function Aggregation capability to identify the subset of functions to focus on and aggregate them into Function Groups.

To do this, select **Advanced > Function Aggregation**.

You can find the functions that are present in the trace file, but not in the function group, in the **Other** group. By default, the data for the **Other** group is displayed. To hide data for the **Other** group on the Quantitative chart, select **Advanced > Show Function Group 'Other'**.

Revert to the default aggregation in **Advanced > Default Aggregation**.

See Also

Function Aggregation

Creating Command Line for Intel® VTune™ Amplifier XE and Intel® Advisor XE

You can create a command line for analyzing specific processes with Intel® VTune™ Amplifier XE or Intel® Advisor XE.

Go to **Advanced > Command line for VTune Amplifier...**, or **Advanced > Command line for Advisor...**

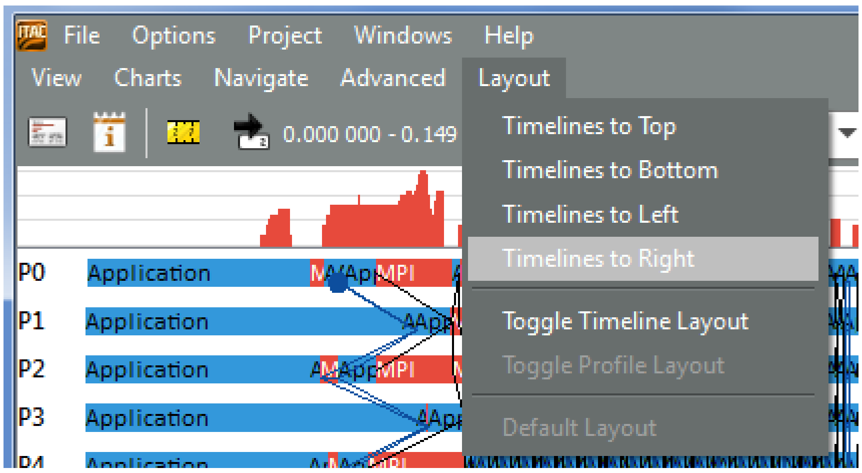
You will see a dialog box with a command line example for process 0. In the upper text box type in the numbers of processes you want to analyze with Intel VTune Amplifier/Intel Advisor. In the text box below you will see the resulting command line. You can copy the command line by clicking the **Copy To Clipboard** button and use it to run Intel VTune Amplifier/Intel Advisor.

See Also

Command line for Intel® VTune™ Amplifier XE and Intel® Advisor XE Dialog Box
Interoperability with Intel® VTune™ Amplifier XE and Intel® Advisor XE

Layout

Generally, every View is split into two sections: Timelines and Profiles. Use the Layout menu to organize the Charts.



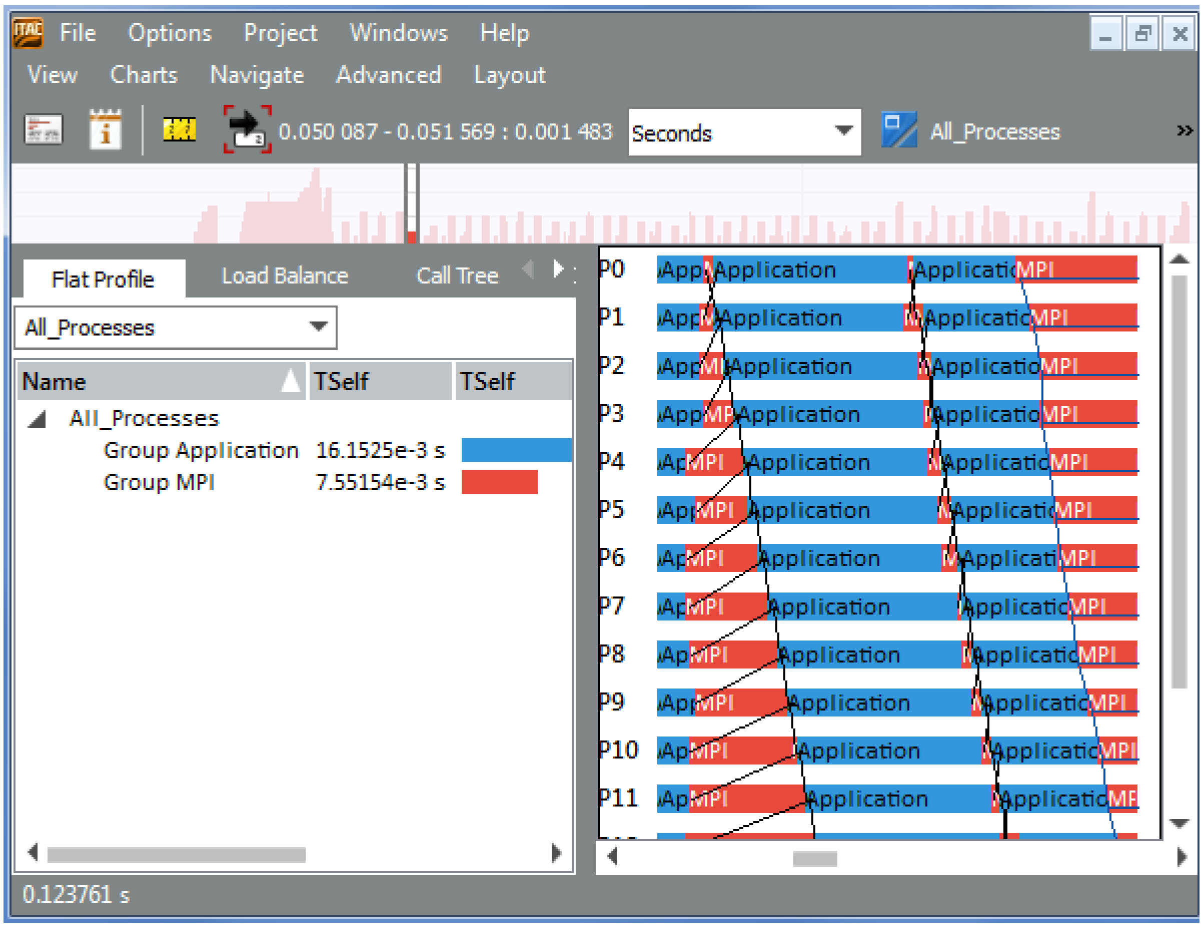
The Layout choices are available when there are two or more charts open.

The following entries are available in the Layout menu:

Option:	**Description:**
Timelines to Top	Move the timelines to the top of the View and the profiles to the bottom.
Timelines to Bottom	Move the timelines to the bottom of the View and the profiles to the top.
Timelines to Left	This moves the timeline(s) being shown in the View to the left of the screen, and also shifts the profiles being shown to the right side. When there are multiple timelines visible, they are collectively shown one below the other on the left side.
Timelines to Right	Place all the timelines to the right of the View. It works the same way as the **Timelines to Left** option.

Toggle Timeline Layout	Change the location of the timelines relative to each other. It is useful only when two or more timelines are open at the same time. If the timelines are stacked and aligned on top of each other, use this option to present them next to each other (or vice versa). The menu entry switches between the two choices.
Toggle Profile Layout	Change the position of the profiles relative to each other, similar to the **Toggle Timeline Layout** option.
Default Layout	Restore the View layout to the default settings. By default, timelines are stacked vertically at the top of the View with Profiles along the bottom, side by side.

For example, select the **Timelines to Right** to place the Event Timeline on the right and the Function Profile on the left:



Comparison Menu

The Comparison feature is available from the Views menu bar only when the Comparison mode is activated. To activate it, select **View > Compare**.

The Comparison menu contains the following entries that enable you to customize the navigation of the compared charts:

- Same Time Scaling
- Synchronize Navigation Keys
- Synchronize Mouse Zoom

See Also

Comparison of two Trace Files

2.2.5. Views

For a flexible analysis of a tracefile, look at multiple partitions of the data from various perspectives using several charts opened in the same View. A View holds a collection of Charts in a single window. The Charts of the same View use one and the same perspective on the data. This perspective is made up of the following attributes:

- Time interval
- Process aggregation
- Function aggregation
- Filters

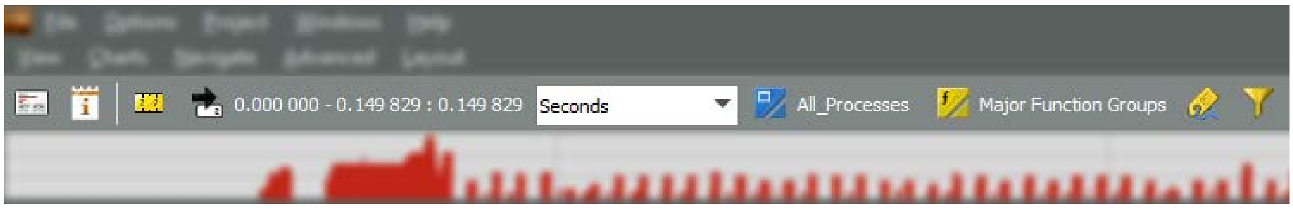
For a detailed explanation of aggregation and filters, refer to Concepts.

A View is a perspective comprised of charts, and whenever you change an attribute for one Chart in the current perspective (in the current View), all other Charts of the same View are changed too.

You can use several Views at the same time, as it is a very flexible and variable mechanism for exploring, analyzing and comparing trace data.

Toolbar

The View window toolbar provides you with easy access to the Intel® Trace Analyzer functionality that is most often used during trace analysis.



The following functionality is available from the toolbar:

Option:	Description:
Summary Page	Open the **Summary Page** to get summary information on the time your application spent in MPI
Tracefile Info	Open **Tracefile Info** to display information about the opened trace
Time Scale	Show Time Scale along the top of all the Timelines. You can enable the Time Scale only if there is at least one timeline opened.
Time Interval Selection	Open **Select Time Interval** to select a new time interval for the whole View. To the right of the **Time Interval Selection** button you can see the data cell that demonstrates the selected time interval in the format: <`start`>, <`end`>: <`duration`>.
Time Units	Switch time units from seconds to ticks and vice versa.
Process Aggregation	Open **Process Aggregation** to select the process group for aggregation or create new process groups in addition to those

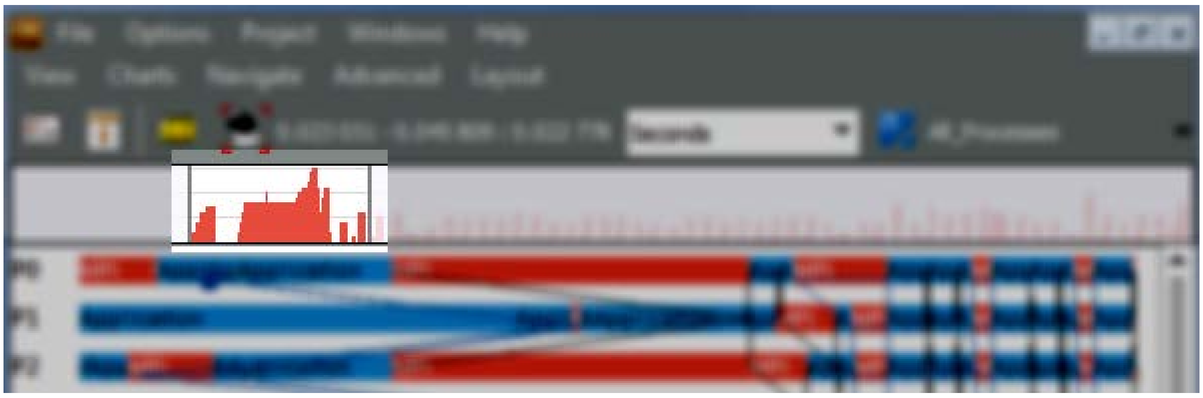
	provided by default. The data cell to the right of the **Process Aggregation** button shows the currently selected process group. By default, it is All_Processes.
Function Aggregation	Open the **Function Aggregation** dialog box and choose a function group for aggregation. The data cell to the right of the **Function Aggregation** button shows the currently selected function group. By default, it is Major Function Groups.
Tagging	Open **Tagging** to identify messages and collective operations to highlight. Intel Trace Analyzer displays the tagged messages and collective operations in bold.
Filtering	Open **Filtering** to select events, messages and collective operations to be filtered. After you set a filter, the Trace Analyzer only displays the data that meets the conditions of the filter operation.
Idealization	Open the **Idealization** to produce an ideal trace.
Compare Traces	Open the **Comparison View** to calculate the exact differences and speedups between two runs or within two ranges of the same run.
Imbalance Diagram	Compare the trace with its idealized one. The Imbalance Diagram helps to understand the application imbalance and estimate a potential application speedup from tuning MPI implementation and/or network upgrades.
Performance Assistant	Use the Performance Assistant tool to discover performance problems of your application.
Color Editor	Edit the colors of functions and function groups.
Preferences	Open the **Preferences** to customize your working environment.

See Also

Time Interval Selection
Process Aggregation
Function Aggregation
Tagging Dialog Box
Filtering Dialog
Trace Idealizer Dialog Box
Comparison of Two Tracefiles
Imbalance Diagram Dialog Box
Performance Assistant
Function Group Color Editor
Preferences
Summary Page

Trace Map

Trace Map enables you to zoom into the relevant subsets of large trace file Charts. It represents a miniature view of the MPI function activity over time. By default, the MPI function activity is shown in red color.



The highlighted part of the Trace Map represents the part of the trace file currently displayed on the opened Chart. This highlighted area - the view area - is updated automatically when you start viewing another time interval of the opened Chart. When the time interval of one of the Charts changes, the Trace Map and all opened Charts are updated automatically.

NOTE

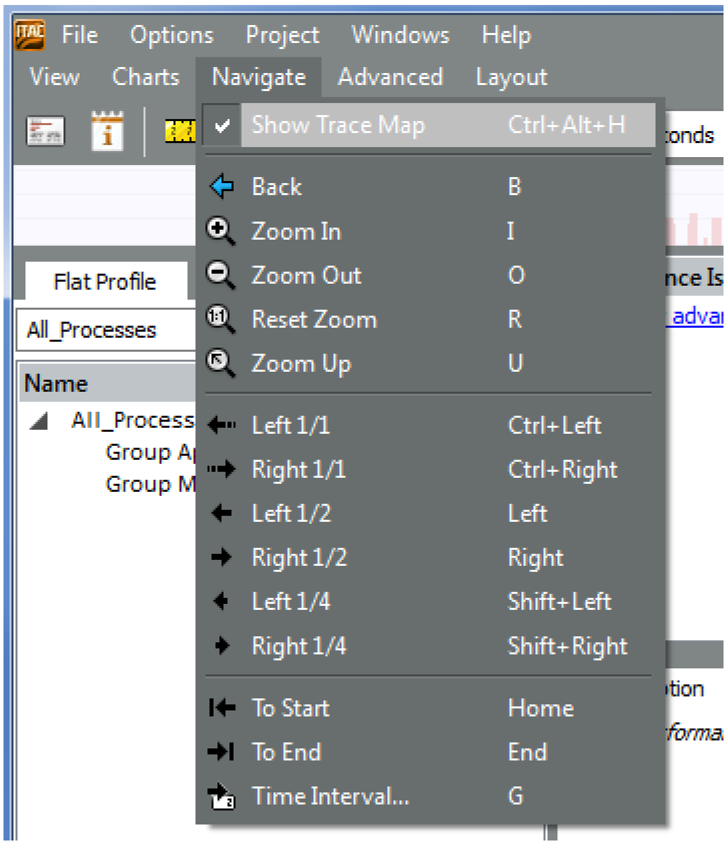
Trace Map is currently unavailable in the Comparison mode.

The Trace Map has a context menu to Print, Save or Hide the Chart. To hide the Trace Map through the View Menu, go to **View Menu** > **Navigation** > **Show Trace Map**.

Navigate a chart with the Trace Map:

Do This:	**To Do This:**
Select a specific area of the Trace Map	Change the time interval and zoom into a specific area of the trace
Press and hold the left mouse button on the selected area of the Trace Map and move the mouse to the right or left	Move the highlighted view area to the right or to the left of the Trace Map and see the desired part of the opened Chart
Use the keyboard shortcut keys listed below	Move, expand or shrink the view area

Shortcut Keys:



See Also

Navigation in Time

Status Bar

When moving the mouse cursor over an event in the Event Timeline and Counter Timeline or an entry in the Message Profile and Collective Operations Profile, the status bar on the bottom border of the View displays expanded information for a few seconds.

2.2.6. Charts

Charts in Intel® Trace Analyzer are graphical or alphanumerical diagrams that are parameterized with a time interval, a process grouping, a function grouping (see Aggregation) and an optional filter. Together they define the structure in which data is presented and the amount of data to be displayed.

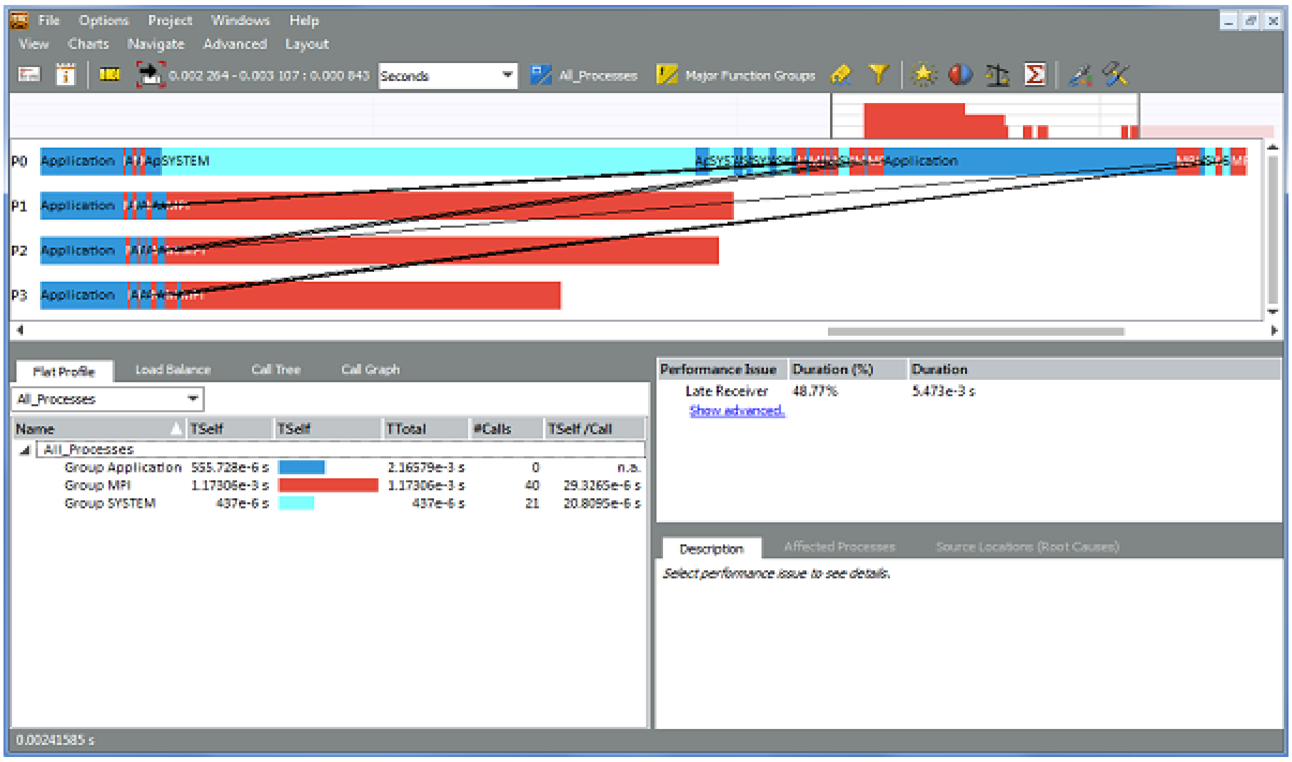
The Charts supported by Intel Trace Analyzer are divided into:

- **Timelines:** Event Timeline, Qualitative Timeline, Quantitative Timeline and Counter Timeline.
- **Profiles:** Function Profile, Message Profile and Collective Operations Profile.

Intel Trace Analyzer charts enable you to see the following data:

- MPI calls
- Application code (includes User Code and OpenMP* regions)
- System Calls

For example, you can see all the three groups of data on the **Event Timeline** and **Flat Profile** of the **Function Profile** chart:



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes					
Group Application	555.728e-6 s	🟦	2.16579e-3 s	0	n.a.
Group MPI	1.17306e-3 s	🟥	1.17306e-3 s	40	29.3265e-6 s
Group SYSTEM	437e-6 s	🟦	437e-6 s	21	20.8095e-6 s

Performance Issue	Duration (%)	Duration
Late Receiver	48.77%	5.473e-3 s

Show advanced.

Select performance issue to see details.

You can ungroup Group `SYSTEM` to see all the traced system functions. Get more detailed information in the Function Profile Chart section.

Ungroup the `Application` group to see the information about user code and OpenMP parallel regions.

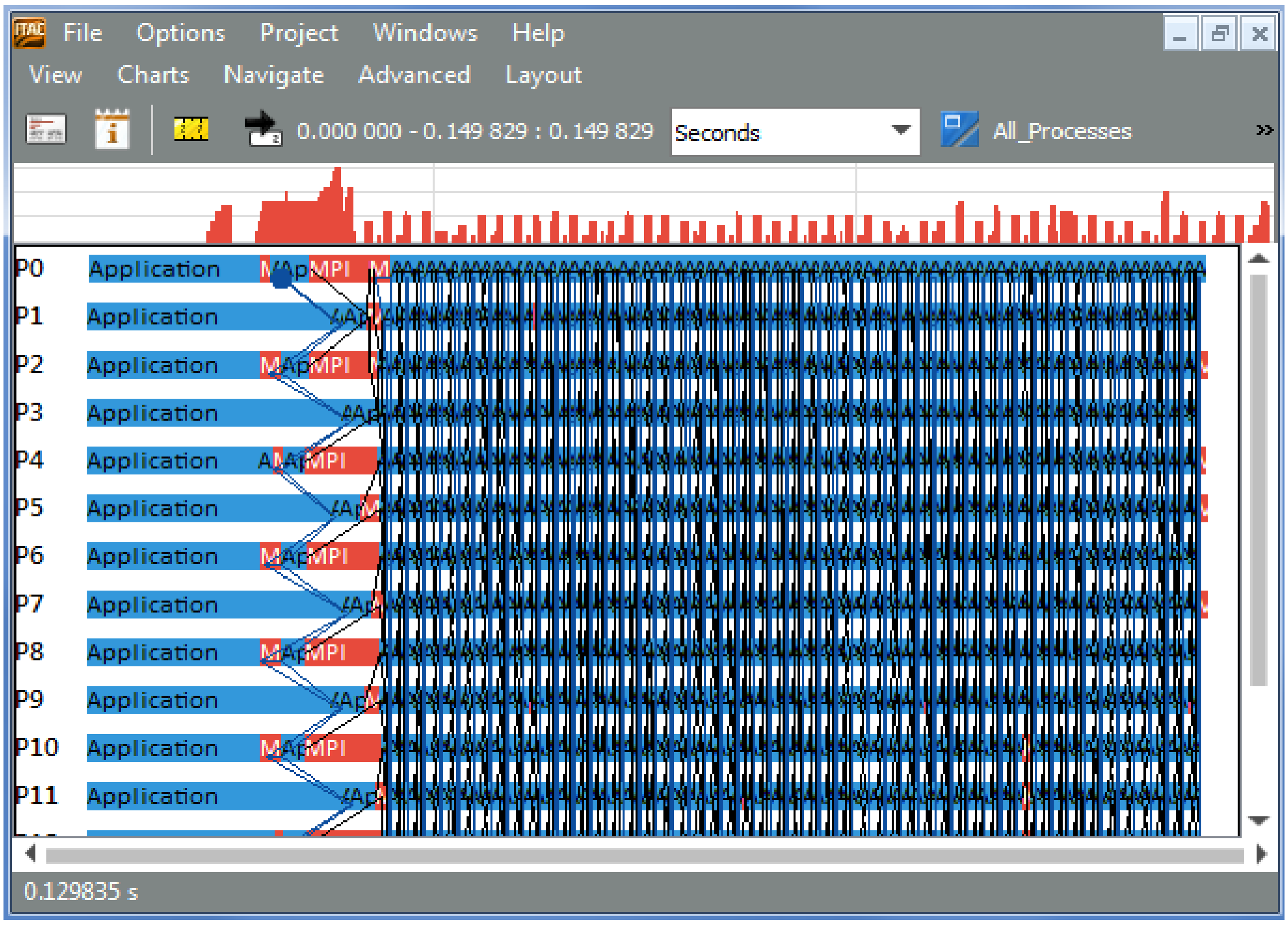
Charts are grouped into Views. These Views provide ways to choose the time interval, the process grouping and optional filters that all Charts in the View use. For more details on Views, see Views.

While the timelines show trace data in graphical form over a horizontal axis representing runtime, the profiles show statistical data. You can find all these Charts under **Views Menu > Charts**. By default, a View containing the Function Profile Chart (showing the **Flat Profile** tab) opens when you press **Next** on the Summary Page.

The following sections describe each Chart in detail. For each Chart there is a subsection about Mouse Hovering, its context menu, how and why to customize a chart (settings), the effects of filtering, and tagging and the effects of aggregation when applicable.

Event Timeline

The Event Timeline provides a graphical display of the individual process activities over time. To open the Event Timeline, go to **Charts > Event Timeline**.



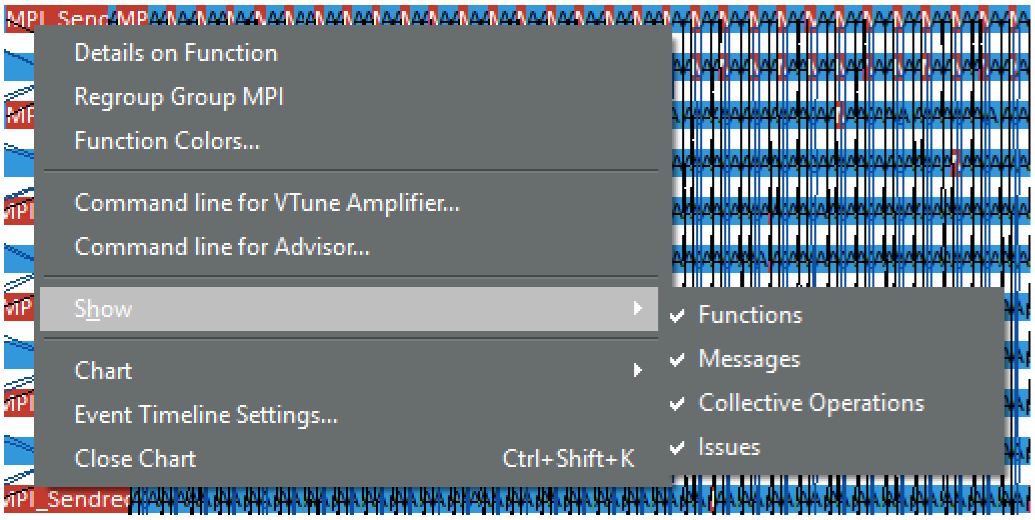
Element:	Description:
Horizontal bars	Represent the processes with the functions called in these processes. The bars consist of colored rectangles labeled with the function names.
Black lines	Indicate messages sent between processes. These lines connect sending and receiving processes.
Blue lines, forming a grid	Represent collective operations, such as broadcast or reduce operations.
Status bar at the bottom of the panel	Shows the entire runtime of the trace. If the mouse cursor hovers over the Event Timeline, the status bar shows information on events under the cursor. To get a better view of events and communications between them, use zooming.

See Also

Event Timeline Settings
Navigate Menu

Context Menu

You can access the context menu by right-clicking on the Event Timeline. Some general context menu options are common to all Charts. Event Timeline provides some options that are specific to this timeline.



Specific Event Timeline context menu options:

Entry:	**Description:**
Details on Function/Message	Access details on a particular function or message in the Event Timeline
Ungroup Group Application/Group MPI	See the OpenMP* regions in the application code or particular MPI functions in the MPI calls
Function Colors...	Open Function Group Color Editor to apply new colors to function groups
Show	Select what elements you want to be displayed: functions, messages or collective operations. By default, all of them are displayed in the Event Timeline.
Chart	Print, save, clone and move the Chart (for more details, see section Common Chart Features)
Event Timeline Settings	Set or reset the Event Timeline preferences
Close Chart	Stop using the Event Timeline
Command line for VTune Amplifier/Advisor...	Open the **Command line for Intel® VTune™ Amplifier XE/Intel® Advisor XE** dialog box for selected process

See Also

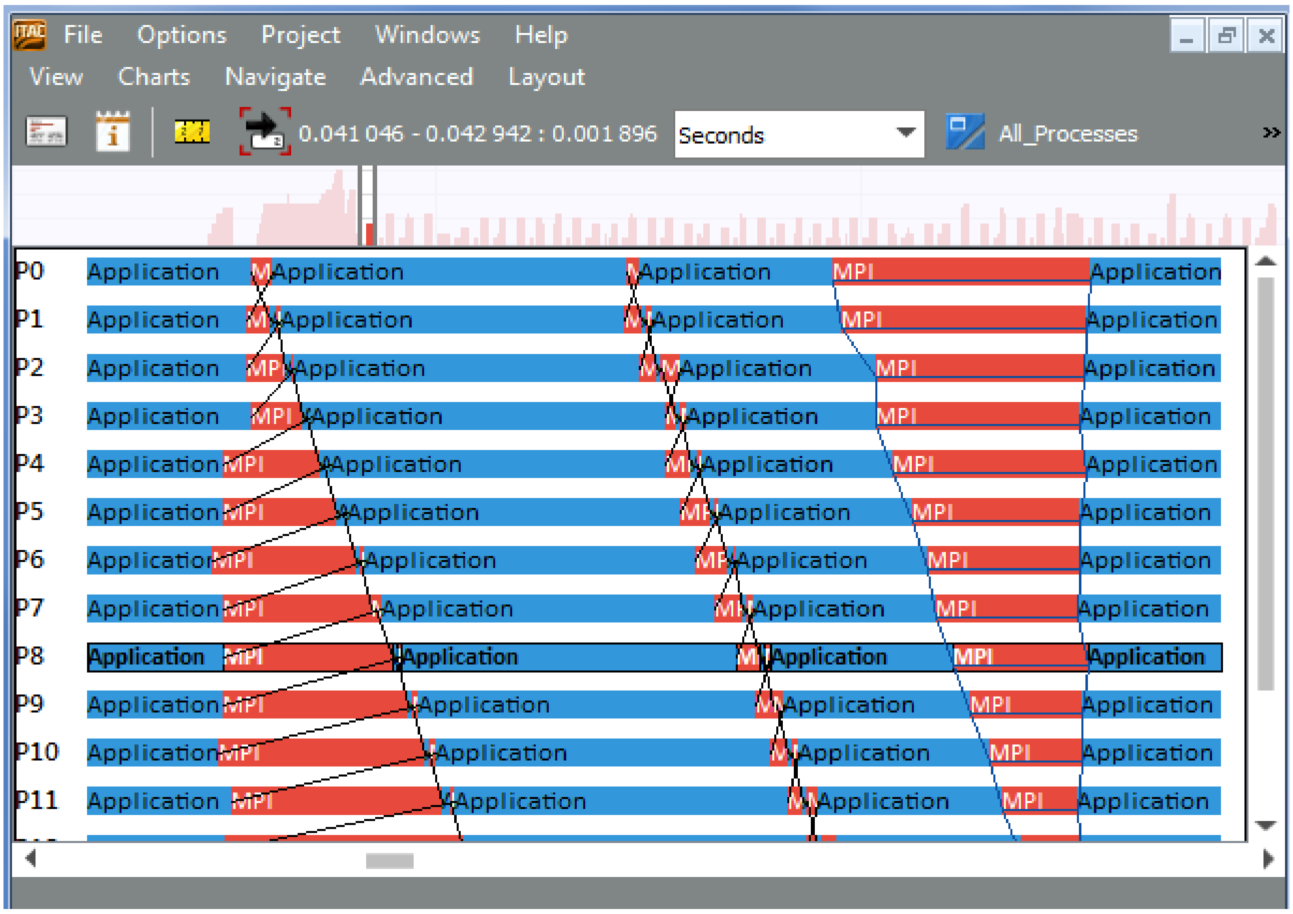
Common Chart Features
Command line for Intel® VTune™ Amplifier XE and Intel® Advisor XE Dialog Box

Filtering and Tagging

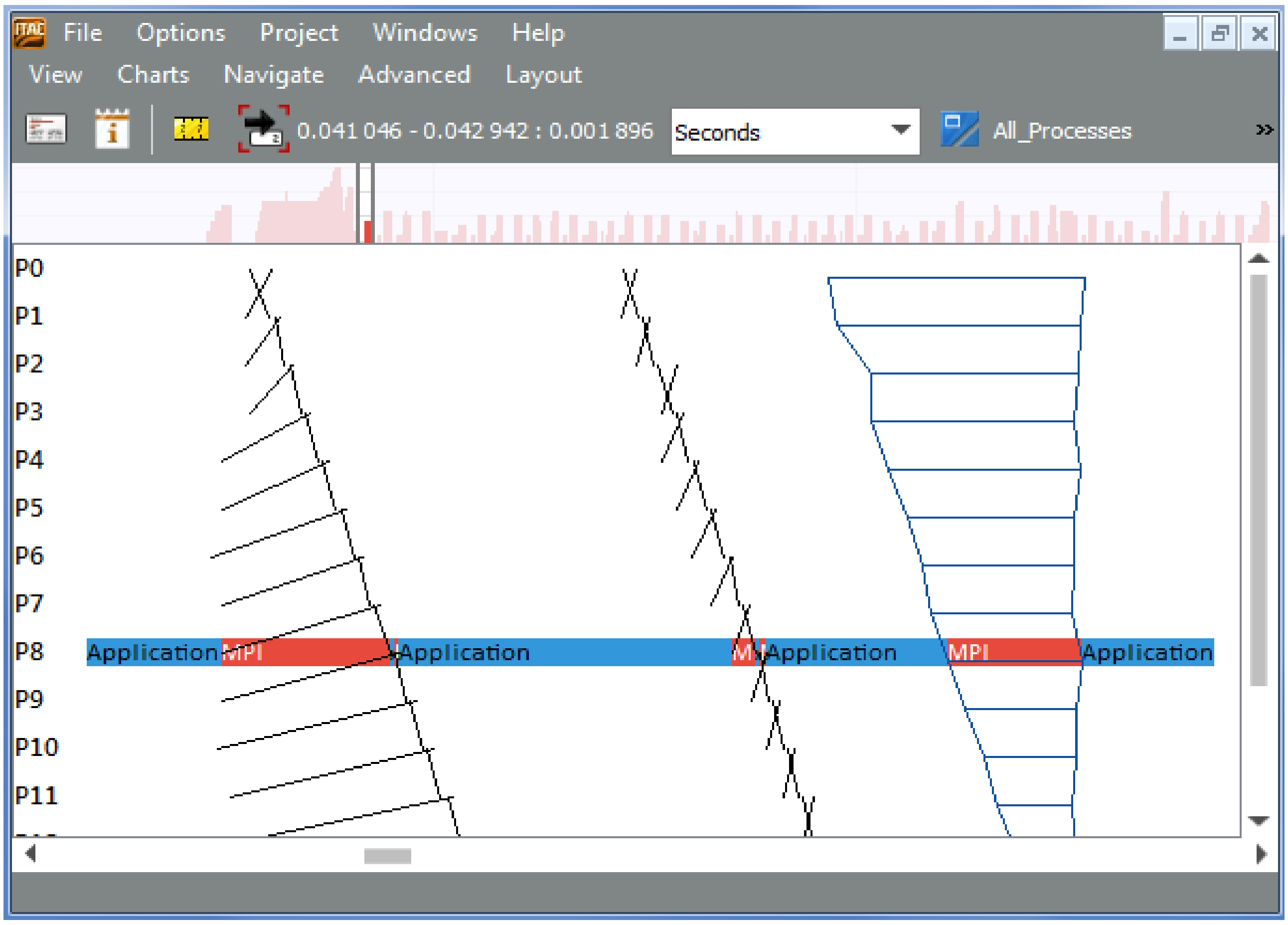
In the Event Timeline, you see the tagged functions in a frame and with their function labels in bold. The lines of the tagged messages and collective operations are thicker.

When you filter the Event Timeline for a particular process, you see only the MPI functions of this process, since only they pass the filter and all the other functions and processes are filtered out.

Tagging Functions in Process 8 in the Event Timeline:



Filtering Functions in Process 8 in the Event Timeline:



See Also

Filtering Dialog Box
Tagging Dialog Box
Tagging and Filtering

Qualitative Timeline

The Qualitative Timeline shows event attributes, such as the data volume of messages, as they occur over time. To open the Qualitative Timeline, select **Charts > Qualitative Timeline**.

The value of the event attribute is plotted along the vertical (y) axis and time is plotted along the horizontal (x) axis. Select the required event type and attributes from the context menu. Use the Qualitative Timeline to detect patterns and irregular behavior such as extreme deviations or long-term changes in attribute values.

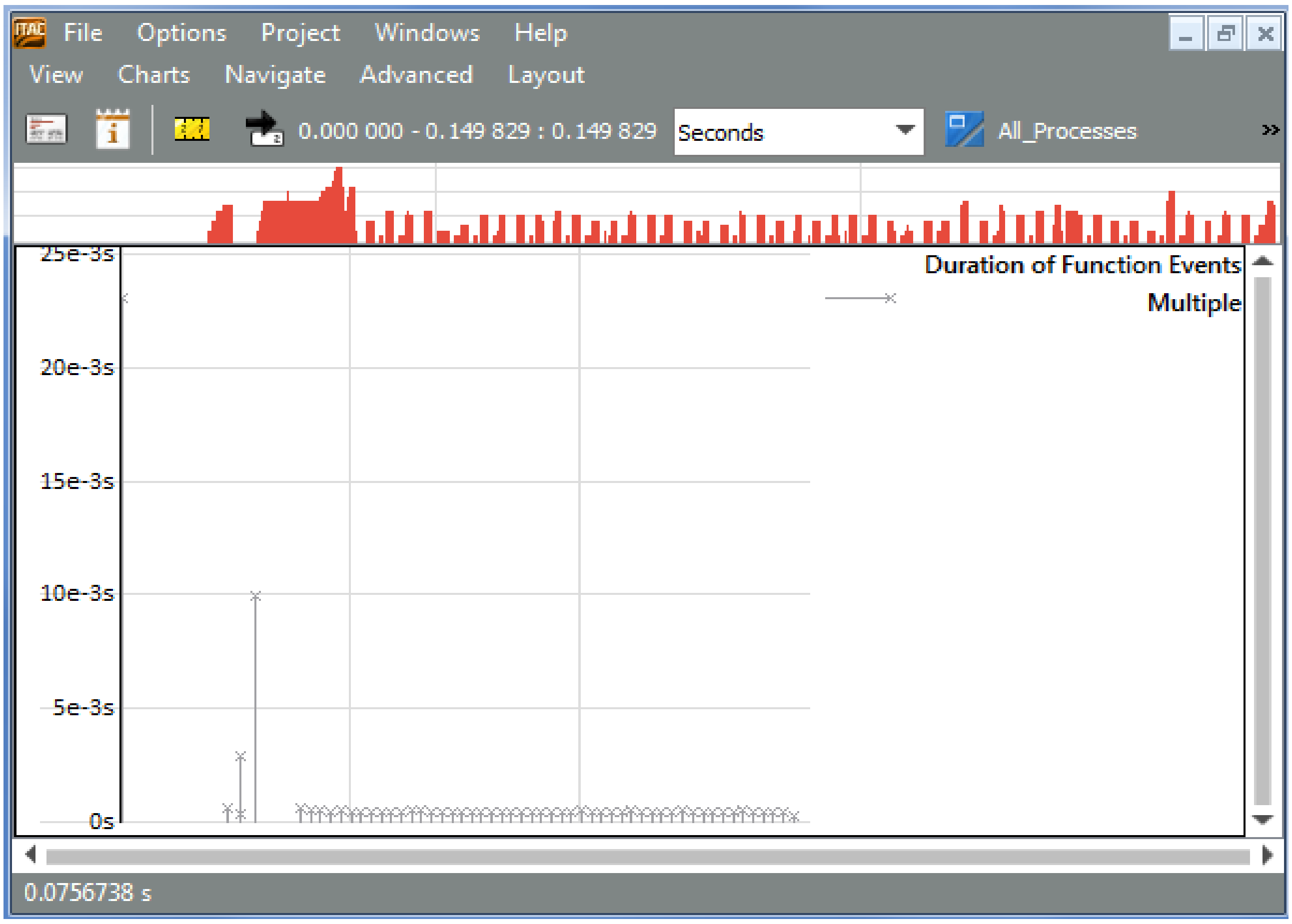
A vertical line in the Qualitative Timeline can represent:

- a single event (denoted as **Single** in the legend)
- several events grouped together (denoted as **Multiple** in the legend)

For the Poisson example, Qualitative Timeline exposes the pattern of function events. To see it, do the following:

1. Open the Qualitative Timeline using **Charts > Qualitative Timeline**.
2. Right-click on the Qualitative Timeline to display its Context Menu. Select **Events to show** and then select **Function Events**.
3. Zoom into a bunch of iterations.

As a result, you see the staircase pattern that is indicative of your application being serialized:



For more information on detecting and removing serialization in your application, refer to the *Tutorial: Detecting and Removing Unnecessary Serialization*.

See Also

Qualitative Timeline Settings
Level of Detail

Context Menu

Access the context menu by right-clicking on the Qualitative Timeline. Some general context menu options are common to all Charts. Qualitative Timeline provides some options that are specific to this timeline.

Specific Qualitative Timeline context menu options:

Entry:	**Description:**
Event to show	Choose the event type from **Function Events**, **Messages** and **Collective Operations**
Attribute to show	Choose the event attribute value from **Duration**, **Transfer Rate** or **Data Volume**

NOTE

Not all attributes are available for all event types.

Filtering and Tagging

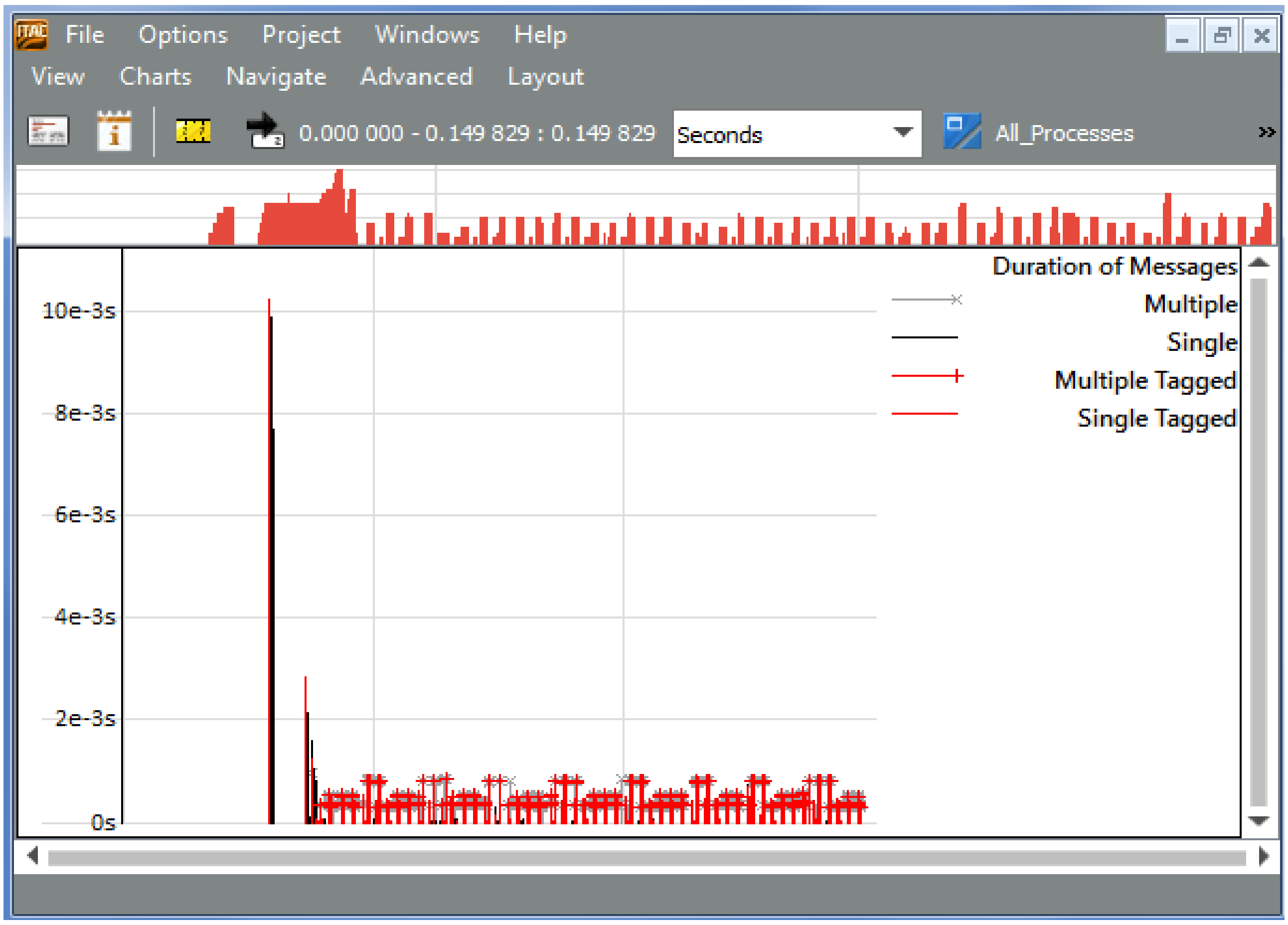
Use tagging in the Qualitative Timeline to find specific events that occur infrequently. You can find these events because a grouped multiple event is tagged if at least one of the singular events it represents matches the tagging filter expression.

Tagged items in the Qualitative Timeline are highlighted with red.

For example, to tag all messages sent by P3 and P13, do the following:

1. Open the **Tagging** dialog box and go to the **Messages** tab.
2. In the group **Messages to be Tagged,** select the radio button **Custom**.
3. Specify whatever needs to be tagged in the given field, for example P3, P13.

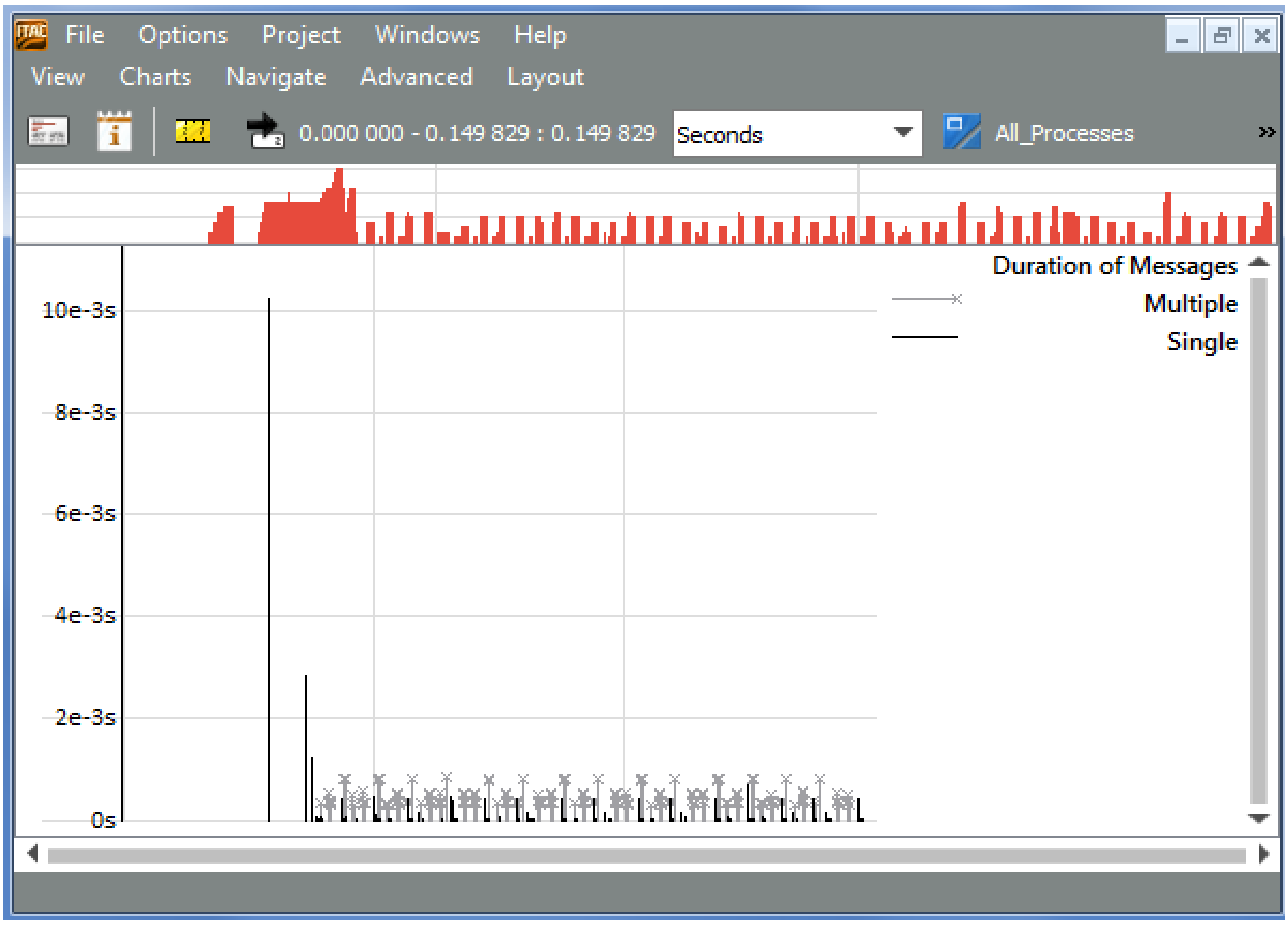
When you tag messages in Processes 3,13 in Qualitative Timeline, the result looks similar to:



When you filter the Qualitative Timeline for a particular event, you see only this event, since only this event passes the filter and all the other events are filtered out.

For example, to filter out all messages except those sent by P3 and P13, use the **Filtering Dialog Box**.

When you filter messages in Processes 3,13 in Qualitative Timeline, the result looks similar to:



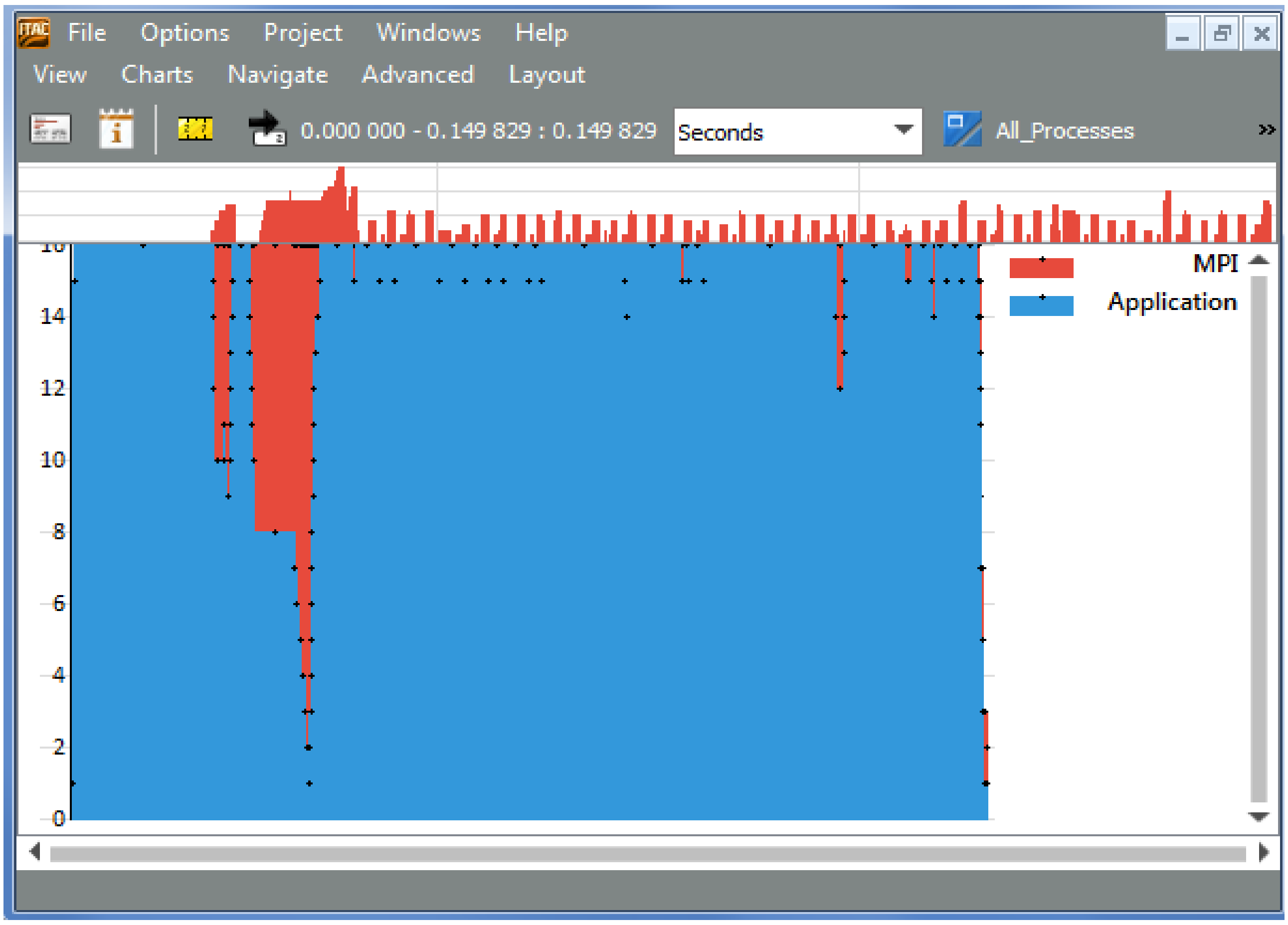
See Also

Filtering Dialog Box
Tagging Dialog Box
Tagging and Filtering

Quantitative Timeline

The Quantitative Timeline gives an overview of the parallel behavior of the application. It shows over time how many processes or threads are involved in which function. To open the Quantitative Timeline, go to **Charts > Quantitative Timeline**.

Along the time axis, different functions are presented as vertically stacked color bars. The height of these bars is proportional to the number of processes that are currently within the respective function.

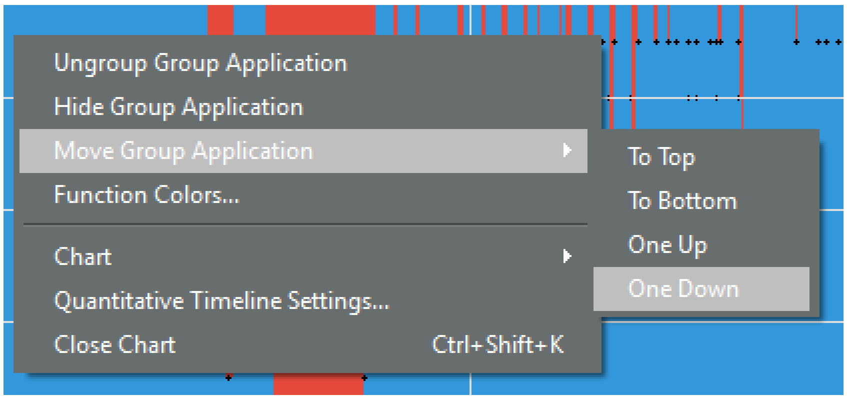


See Also

Quantitative Timeline Settings

Context Menu

You can access the context menu by right-clicking the Quantitative Timeline. Some general context menu options are common to all Charts. Quantitative Timeline provides some options that are specific to this timeline.



Specific Event Timeline context menu options:

Entry:	Description:
Ungroup Group Application/Group MPI	Ungroup the given function group
Regroup Group MPI	Regroup previously ungrouped function group

Hide Group Application/Group MPI	Conceal the chosen activity
Move Group Application/Group MPI	Change the position of different groups The selected group can be moved to the top, to the bottom, upward by one position or downward by one position.
Function Colors...	Open Function Group Color Editor to apply new colors to function groups
Charts	Print, save, clone and move the Chart (for more details, see section Common Chart Features)
Quantitative Timeline Settings	Set or reset the Quantitative Timeline preferences
Close Chart	Stop using the Quantitative Timeline

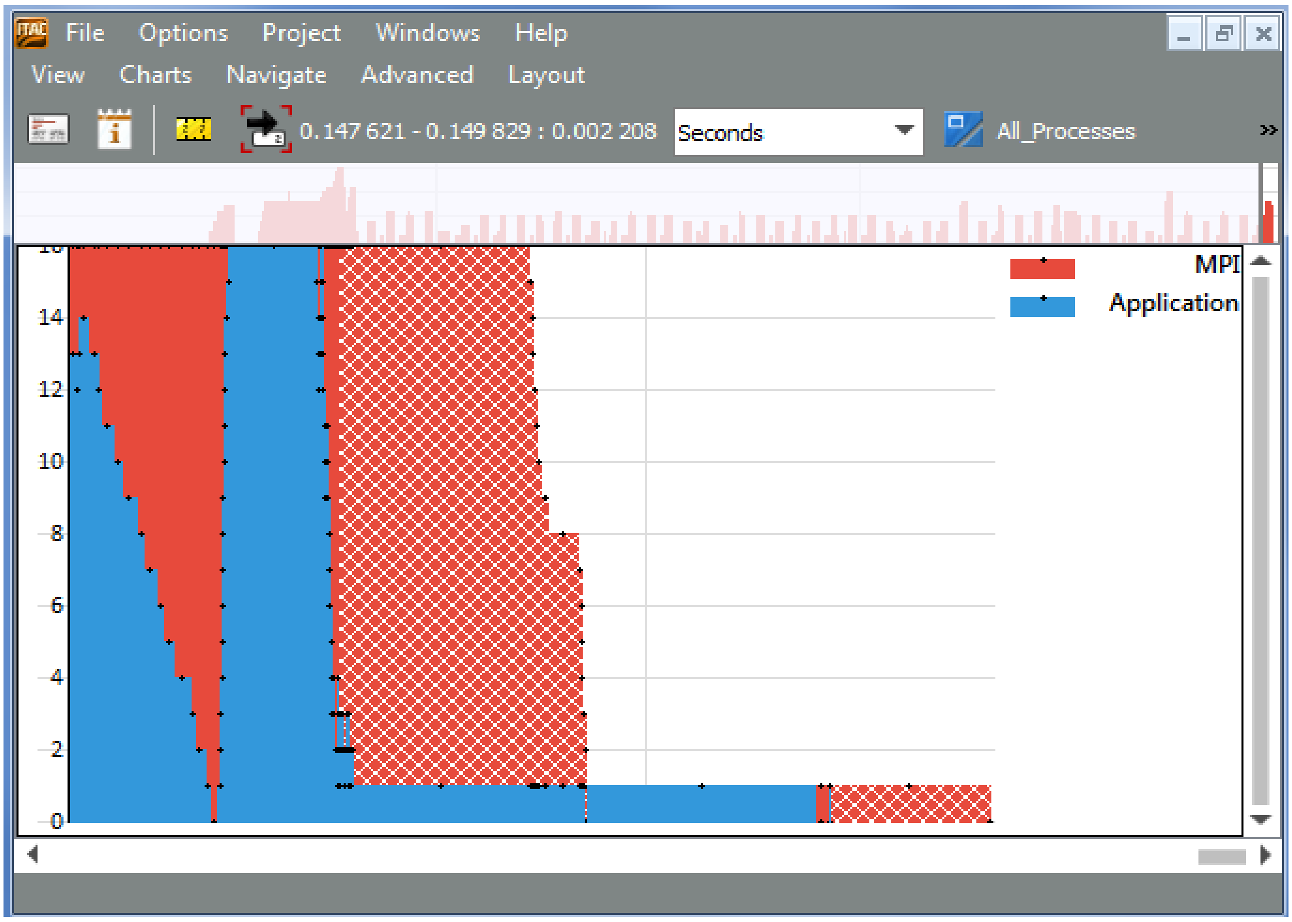
See Also

Common Chart Features

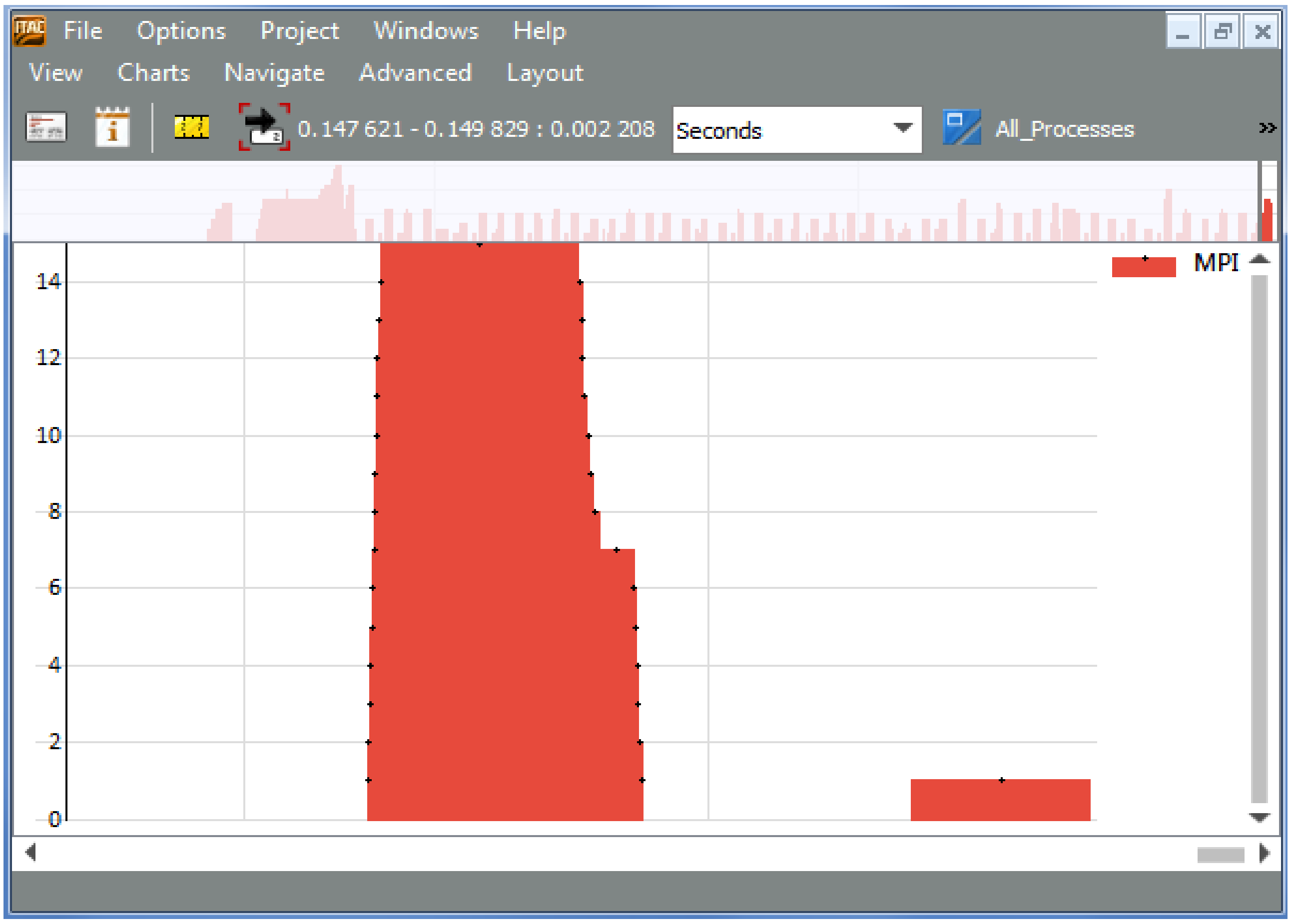
Filtering and Tagging

A mesh pattern indicates tagged items in the Quantitative Timeline.

This is what the Quantitative Timeline for the `poisson_icomm.single.stf` trace file with the tagged `MPI_Finalize` function looks like:

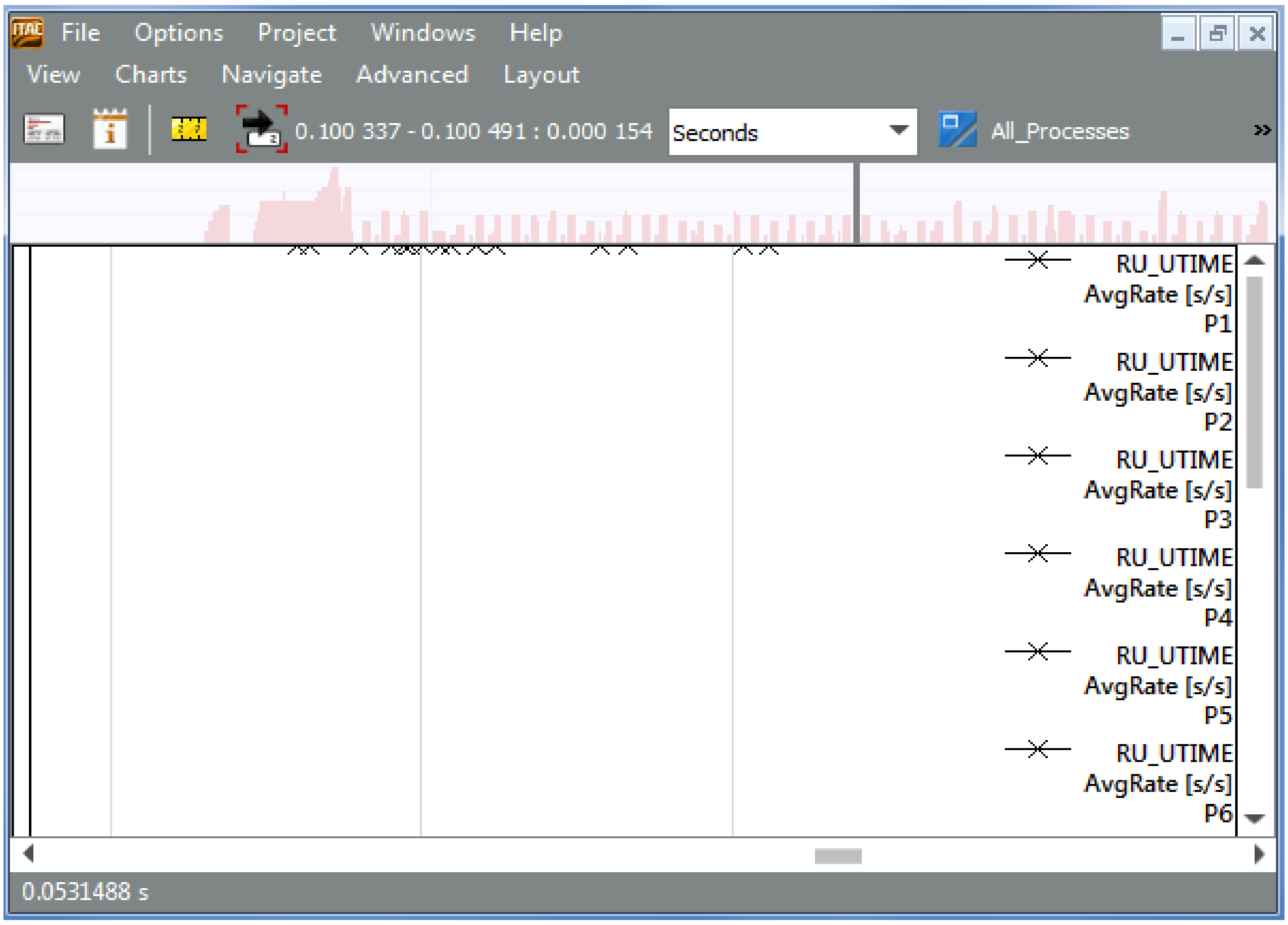


Filtering in the Quantitative Timeline works the same way as in any other Chart. The result for filtering `MPI_Finalize` looks like:



Counter Timeline

The Counter Timeline shows the values of all counters that a given trace file provides. The Intel® Trace Collector records user-defined counters or counters provided by the operating system. See the *Intel® Trace Collector User and Reference Guide* to learn more about tracing counters.


File Options Project Windows Help
View Charts Navigate Advanced Layout
0.100 337 - 0.100 491 : 0.000 154
Seconds
All_Processes
RU_UTIME AvgRate [s/s] P1
RU_UTIME AvgRate [s/s] P2
RU_UTIME AvgRate [s/s] P3
RU_UTIME AvgRate [s/s] P4
RU_UTIME AvgRate [s/s] P5
RU_UTIME AvgRate [s/s] P6
0.0531488 s

Element:	**Description:**
Line segments	Connect sample points for counters with the Curve attribute
Markers	Show sample points with the Valid AT point attribute
Short unconnected horizontal line segments	Show counter samples with other scopes
Status bar at the bottom of the panel	If the mouse pointer is near a sample point, then its exact position in time is shown.

NOTE

There are no markers anymore because the values shown represent samples that were merged over time. To force that effect the View was made very narrow.

See Also

Counter Timeline Settings
Level of Detail

Context Menu

The Counter Timeline context menu provides the common entries as defined in Common Chart Features.

Filtering and Tagging

The filter mechanism in the Intel® Trace Analyzer does not cover counters. Therefore the Counter Timeline is independent of the View filter settings.

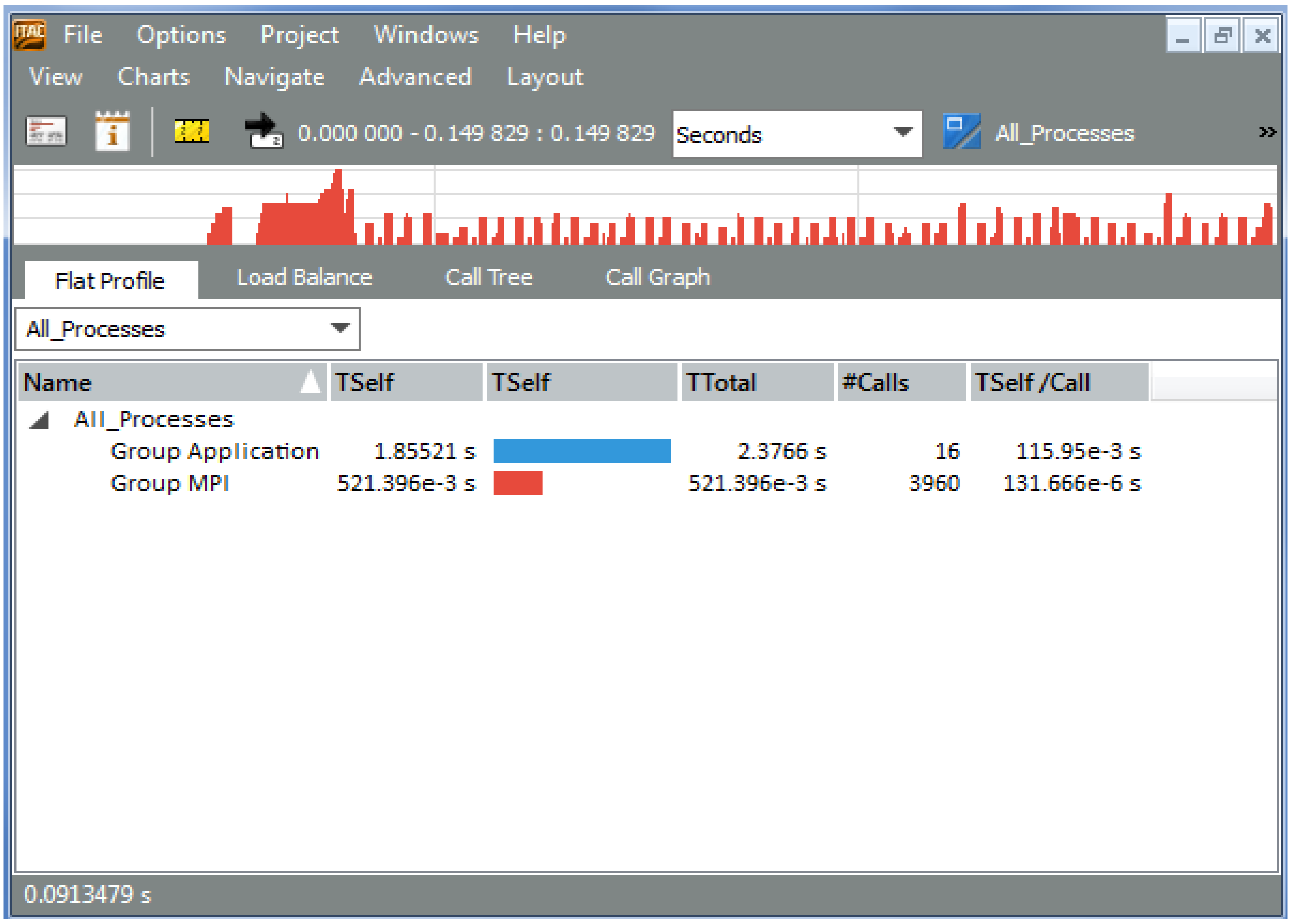
Function Profile

The Function Profile provides detailed profiling information on the performance data.

The Function Profile consists of four tabs: **Flat Profile**, **Load Balance**, **Call Tree** and **Call Graph**. These tabs use the same column headers and the same raw data. The default column headers on display are Name, TSelf, TTotal, #Calls, and TSelf/Call. For a detailed explanation of all available columns refer to the Function Profile Settings section.

NOTE

The functions in the columns are not in the alphabetical order. You see them in the order given by the layout of the current process or function group.



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes					
Group Application	1.85521 s	🟦	2.3766 s	16	115.95e-3 s
Group MPI	521.396e-3 s	🟥	521.396e-3 s	3960	131.666e-6 s

0.0913479 s

Do This:	**To Do This:**
Drag column headings	Adjust the column order
Click on a column header	Sort a list in the ascending or descending order. The arrow symbol in the column header indicates whether the entries are arranged in the ascending or descending order.
Click **TSelf**	See which process spends the most time (or the least time) in a function
Go to **Options > Preferences > Number Formatting**	Set number formatting: increase or decrease the number of digits in numbers shown.

See Also

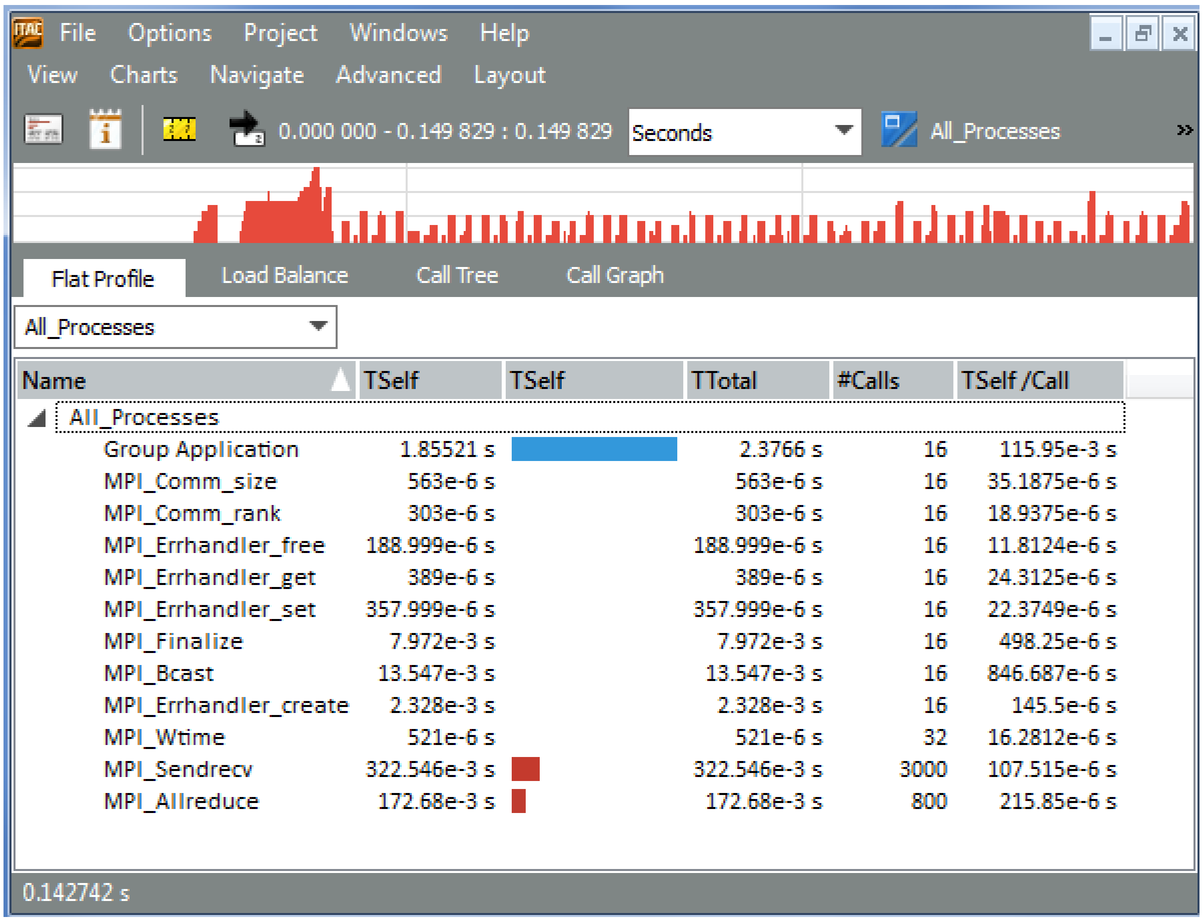
Function Profile Settings
Number Formatting Settings

Flat Profile

By default, the Flat Profile summarizes all major groups of functions and presents statistics over the processes. The exact contents of these groups depend on the group definitions stored in the trace file or you can define them by yourself. In the file `poisson_icomm.single.stf`, these are only MPI and Application groups.

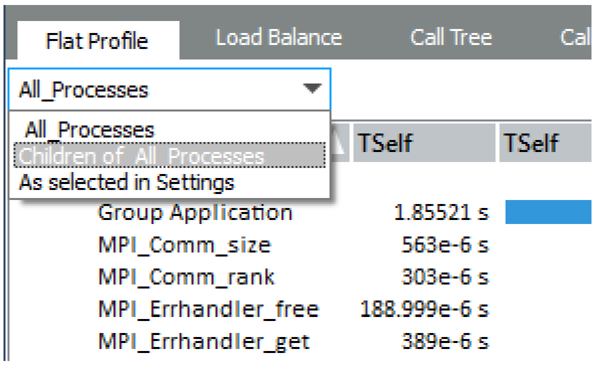
Do This:	To Do This:
Right-click on **Group MPI** and select **Ungroup MPI** from the context menu	See the distribution of execution time over the individual MPI routines. For example, see the Ungrouped MPI image.
Right-click on the child of **Group MPI** and select **Regroup MPI** from the context menu. Or go to Function Aggregation (**Advanced > Function Aggregation**, or the toolbar button) and select Major Function Groups	Regroup the children of MPI
Select the **Children of Group All_Processes** entry from the combo box at the top of the tab. See the Selecting Profiles per Process image.	View the data for the children of each process. The result looks similar to the Showing Children of All_Processes image.
Press the arrows at the side of each process in the **Children of Group All_Processes** view	Expand and collapse the processes of interest

Ungrouped MPI:



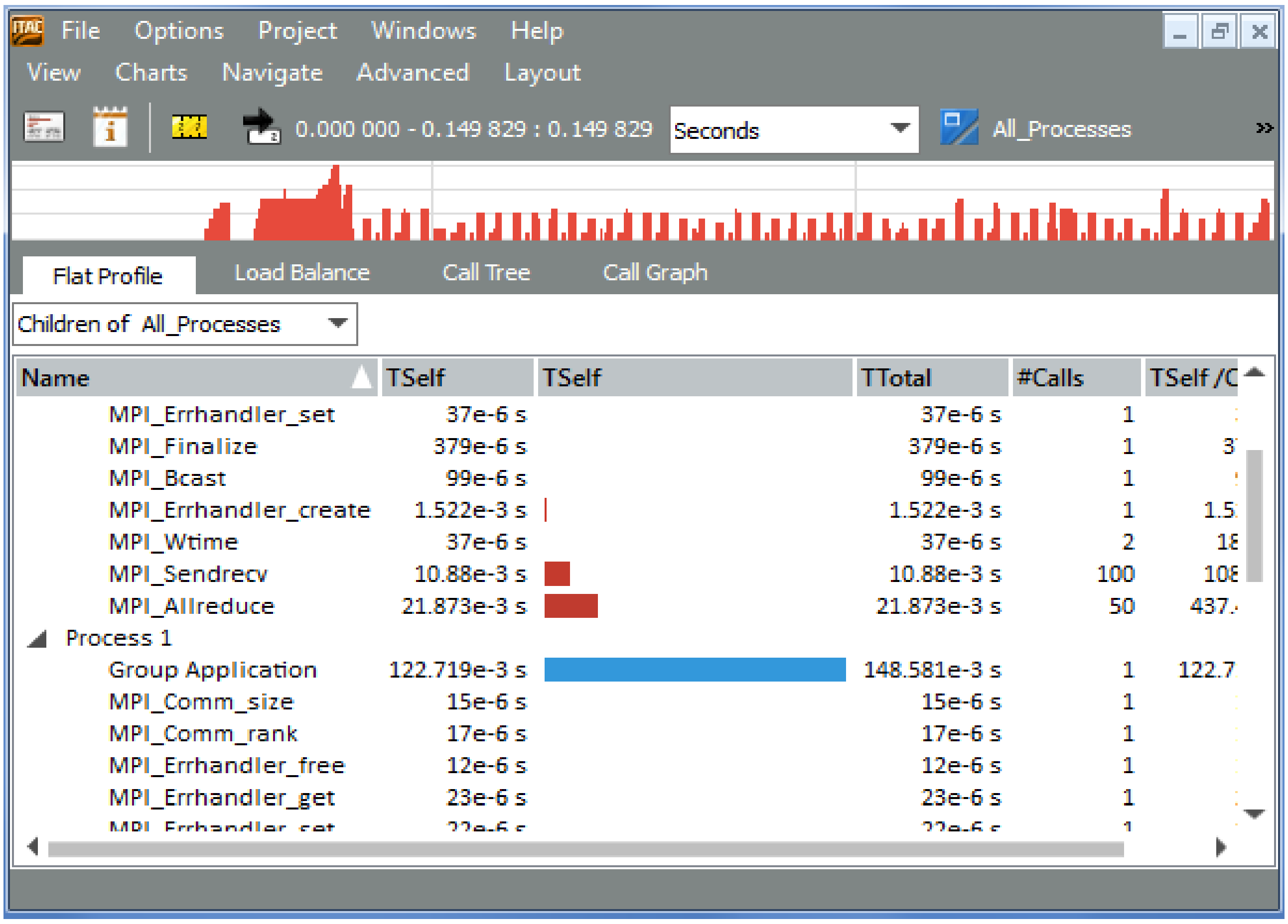
Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes					
Group Application	1.85521 s	🟦	2.3766 s	16	115.95e-3 s
MPI_Comm_size	563e-6 s		563e-6 s	16	35.1875e-6 s
MPI_Comm_rank	303e-6 s		303e-6 s	16	18.9375e-6 s
MPI_Errhandler_free	188.999e-6 s		188.999e-6 s	16	11.8124e-6 s
MPI_Errhandler_get	389e-6 s		389e-6 s	16	24.3125e-6 s
MPI_Errhandler_set	357.999e-6 s		357.999e-6 s	16	22.3749e-6 s
MPI_Finalize	7.972e-3 s		7.972e-3 s	16	498.25e-6 s
MPI_Bcast	13.547e-3 s		13.547e-3 s	16	846.687e-6 s
MPI_Errhandler_create	2.328e-3 s		2.328e-3 s	16	145.5e-6 s
MPI_Wtime	521e-6 s		521e-6 s	32	16.2812e-6 s
MPI_Sendrecv	322.546e-3 s	🟥	322.546e-3 s	3000	107.515e-6 s
MPI_Allreduce	172.68e-3 s	🟥	172.68e-3 s	800	215.85e-6 s

Selecting Profiles per Process:



Name	TSelf	TSelf
Group Application	1.85521 s	🟦
MPI_Comm_size	563e-6 s	
MPI_Comm_rank	303e-6 s	
MPI_Errhandler_free	188.999e-6 s	
MPI_Errhandler_get	389e-6 s	

Showing Children of All_Processes:



Name	TSelf	TSelf	TTotal	#Calls	TSelf /C
MPI_Errhandler_set	37e-6 s		37e-6 s	1	
MPI_Finalize	379e-6 s		379e-6 s	1	3
MPI_Bcast	99e-6 s		99e-6 s	1	
MPI_Errhandler_create	1.522e-3 s		1.522e-3 s	1	1.5
MPI_Wtime	37e-6 s		37e-6 s	2	18
MPI_Sendrecv	10.88e-3 s		10.88e-3 s	100	108
MPI_Allreduce	21.873e-3 s		21.873e-3 s	50	437.
Process 1					
Group Application	122.719e-3 s		148.581e-3 s	1	122.7
MPI_Comm_size	15e-6 s		15e-6 s	1	
MPI_Comm_rank	17e-6 s		17e-6 s	1	
MPI_Errhandler_free	12e-6 s		12e-6 s	1	
MPI_Errhandler_get	23e-6 s		23e-6 s	1	
MPI_Errhandler_set	22e-6 s		22e-6 s	1	

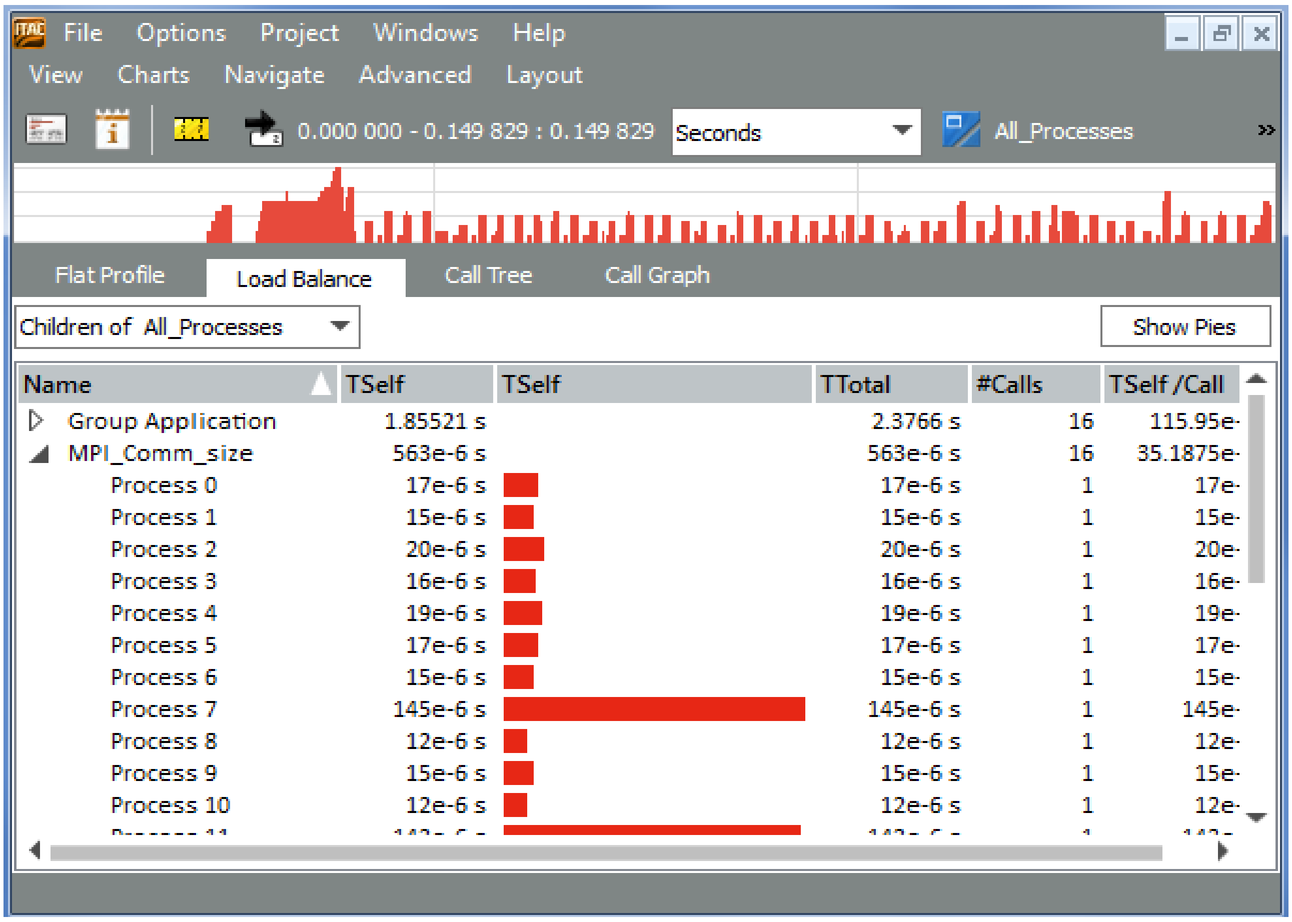
See Also

Function Profile Chart

Load Balance

The **Load Balance** tab displays the same data as the Flat Profile except that it groups the data by function and not by process. The **Load Balance** tab compares the profiles of the same function across several processes. The top level entries of the tree given in the first column are functions.

In this figure showing the load balance for MPI_Allreduce, you can see that TSelf for MPI_Allreduce is pretty unbalanced across processes:

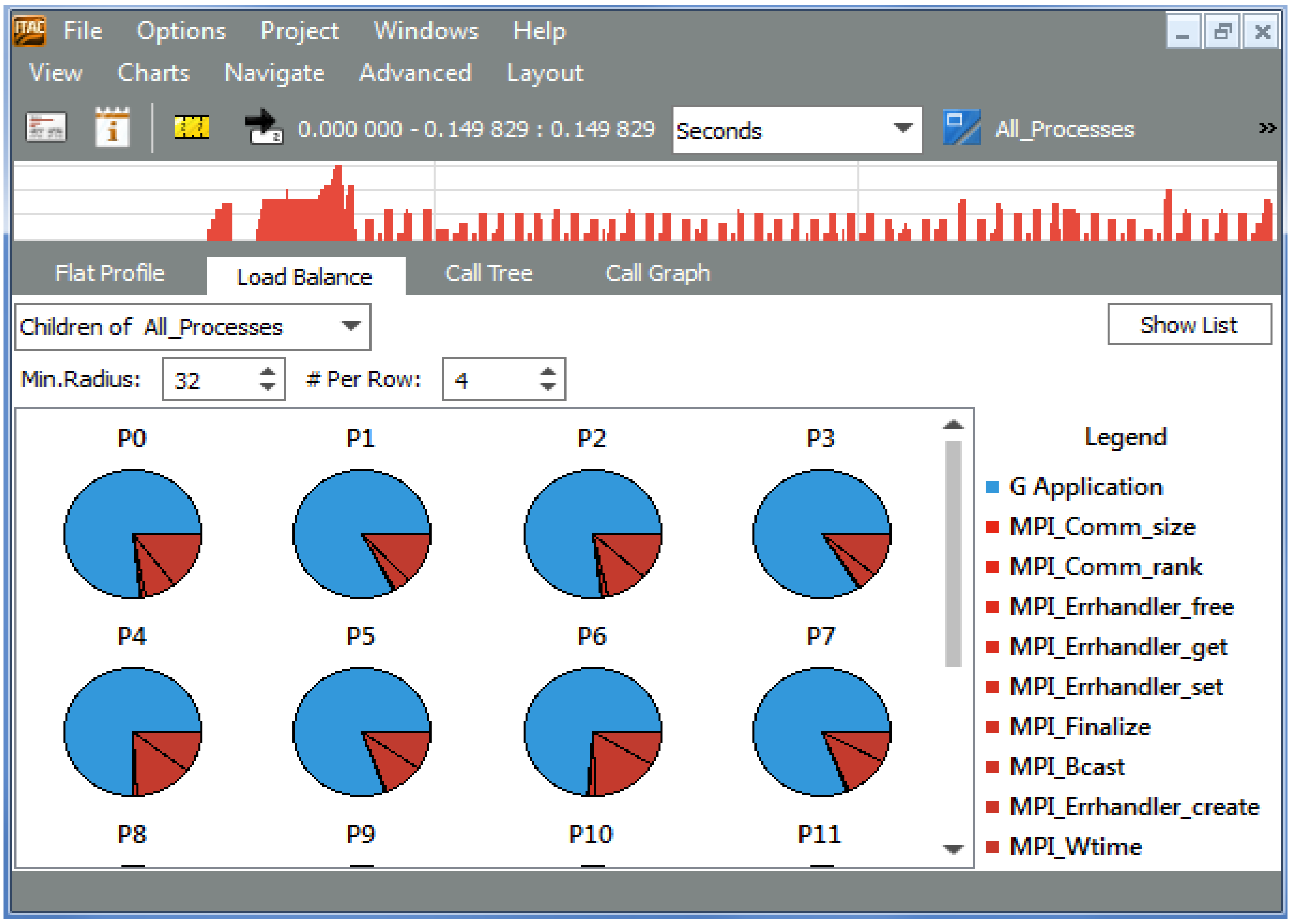


Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group Application	1.85521 s		2.3766 s	16	115.95e-
MPI_Comm_size	563e-6 s		563e-6 s	16	35.1875e-
Process 0	17e-6 s	🟥	17e-6 s	1	17e-
Process 1	15e-6 s	🟥	15e-6 s	1	15e-
Process 2	20e-6 s	🟥	20e-6 s	1	20e-
Process 3	16e-6 s	🟥	16e-6 s	1	16e-
Process 4	19e-6 s	🟥	19e-6 s	1	19e-
Process 5	17e-6 s	🟥	17e-6 s	1	17e-
Process 6	15e-6 s	🟥	15e-6 s	1	15e-
Process 7	145e-6 s	🟥	145e-6 s	1	145e-
Process 8	12e-6 s	🟥	12e-6 s	1	12e-
Process 9	15e-6 s	🟥	15e-6 s	1	15e-
Process 10	12e-6 s	🟥	12e-6 s	1	12e-

Do This:	**To Do This:**
Right-click on **Group MPI** and select **Ungroup MPI** from the context menu	See the distribution of execution time over the individual MPI routines.
Right-click on the child of **Group MPI** and select **Regroup MPI** from the context menu. Or go to Function Aggregation (**Advanced > Function Aggregation**, or the toolbar button) and select Major Function Groups	Regroup the children of MPI
Select the **Children of Group All_Processes** entry from the combo box at the top of the tab.	View the data for the children of each process.
Press the arrows at the side of each process in the **Children of Group All_Processes** view	Expand and collapse the processes of interest
Right-click on a process and select **Command line for VTune Amplifier/Advisor...**	Open the **Command line for Intel® VTune™ Amplifier XE/Intel® Advisor XE** dialog box for selected process
Switch between the list and pie charts by pressing the button in the top right corner of the tab	Analyze the overall load balance pattern (for **TSelf**). For example, see the *Pie Charts in the Load Balance Tab* image. However, there may be a huge number of processes in a relatively confined space.

Use two spin buttons above the pie charts	Control the minimum radius of the pies (left button) and how many pie charts appear in a row (right button)

Pie Charts in the Load Balance Tab



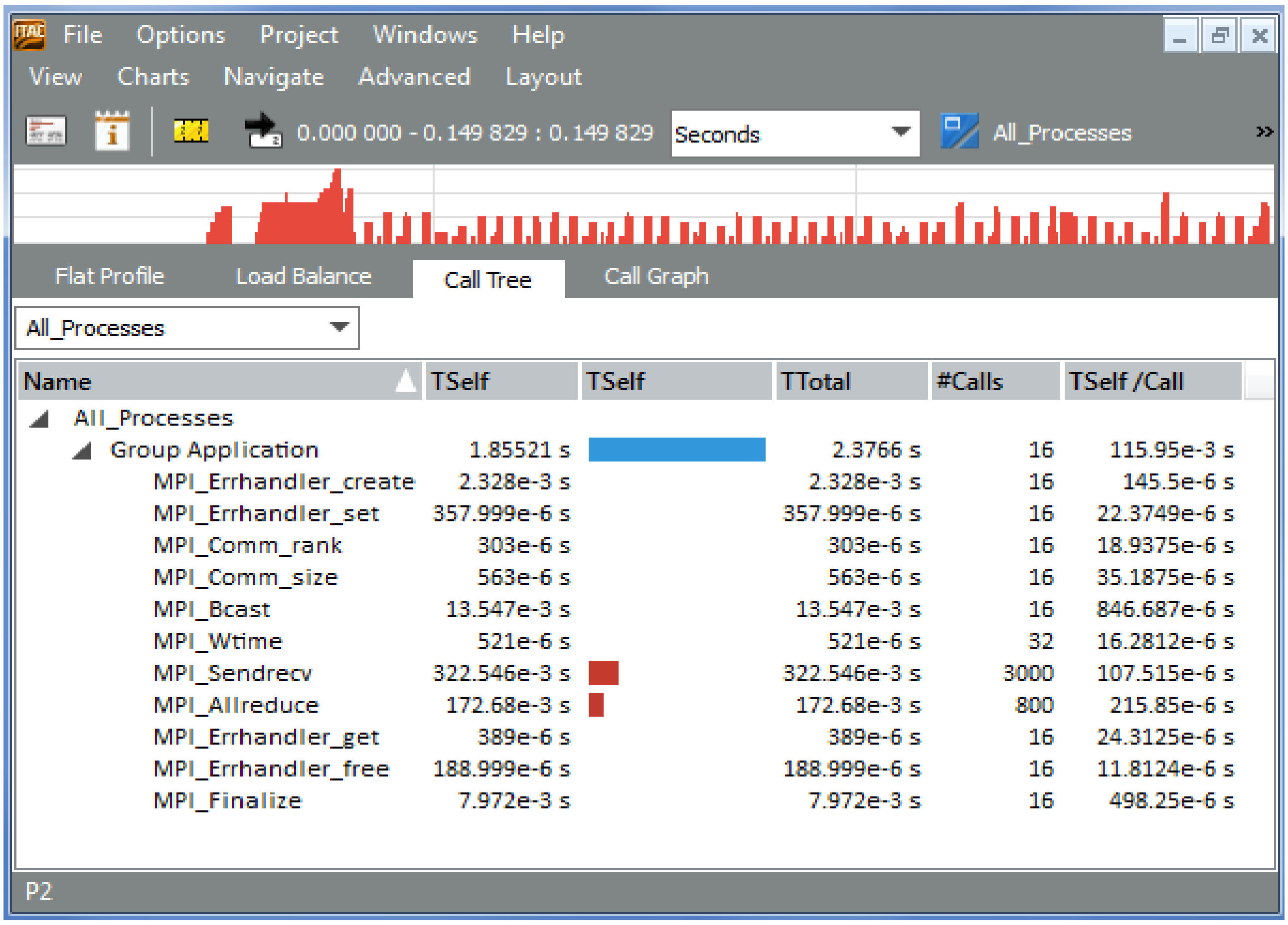
See Also

Command line for Intel® VTune™ Amplifier XE and Intel® Advisor XE Dialog Box

Call Tree

When the **Load Balance** and **Flat Profile** tabs show less detail than you need, use the **Call Tree** tab to analyze calling dependencies. The **Call Tree** demonstrates the calling hierarchy.

In the **Call Tree**, select an entry to focus on. If you change the time interval scrolling or zooming, the focus remains on this entry. The time interval stays selected and visible when possible. If there is no corresponding entry for the new time interval, the **Call Tree** shows its parent. This feature is useful in large and deeply nested call trees.



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes					
Group Application	1.85521 s		2.3766 s	16	115.95e-3 s
MPI_Errhandler_create	2.328e-3 s		2.328e-3 s	16	145.5e-6 s
MPI_Errhandler_set	357.999e-6 s		357.999e-6 s	16	22.3749e-6 s
MPI_Comm_rank	303e-6 s		303e-6 s	16	18.9375e-6 s
MPI_Comm_size	563e-6 s		563e-6 s	16	35.1875e-6 s
MPI_Bcast	13.547e-3 s		13.547e-3 s	16	846.687e-6 s
MPI_Wtime	521e-6 s		521e-6 s	32	16.2812e-6 s
MPI_Sendrecv	322.546e-3 s		322.546e-3 s	3000	107.515e-6 s
MPI_Allreduce	172.68e-3 s		172.68e-3 s	800	215.85e-6 s
MPI_Errhandler_get	389e-6 s		389e-6 s	16	24.3125e-6 s
MPI_Errhandler_free	188.999e-6 s		188.999e-6 s	16	11.8124e-6 s
MPI_Finalize	7.972e-3 s		7.972e-3 s	16	498.25e-6 s

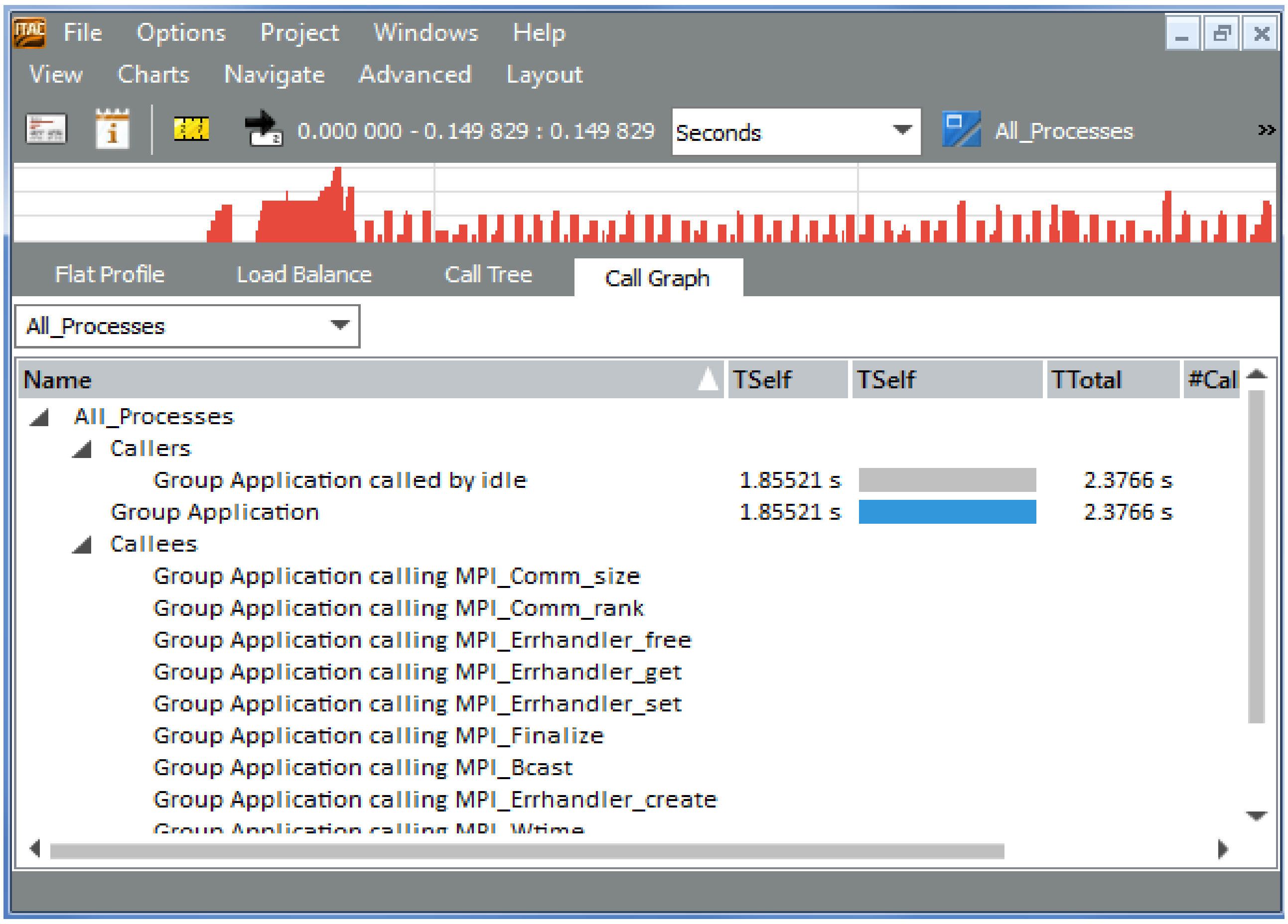
Call Graph

The **Call Graph** tab shows a small part of the call graph for each process or process group: a single node (called central function) with its inbound and outbound edges. Each process entry has three children: the Callers, the central function, and the Callees.

Navigating through the **Call Graph**, double-click on a caller or callee and press the space bar or the Enter key to select the respective function as the central node.

The time shown for the central function is the same as shown in the **Flat Profile** tab and the **Load Balance** tab. The time shown for the callers represents the time spent in the central function when called from the respective function.

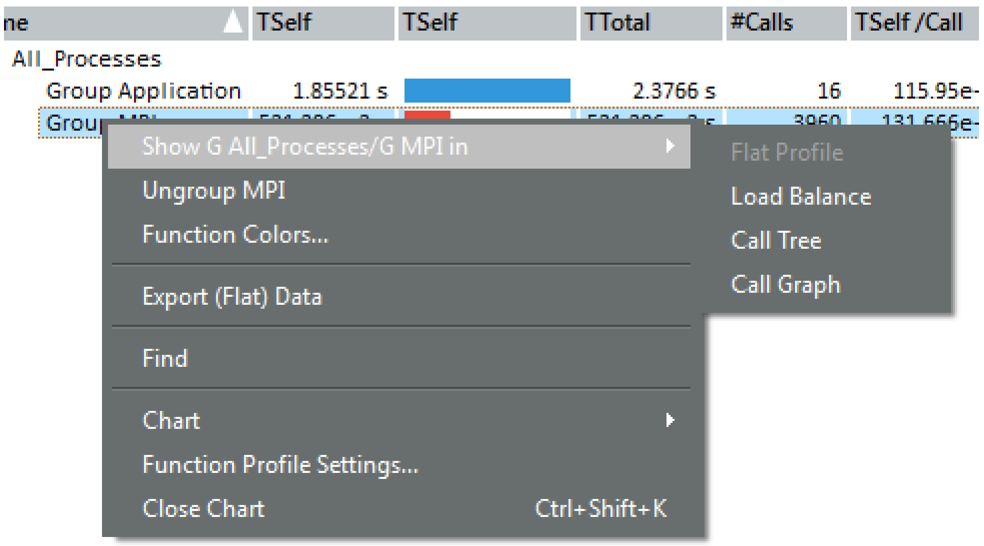
If your application uses a function in different contexts (for example, different algorithms use the function), you can observe which algorithm causes a function to consume more or less time. In the **Call Graph**, you can see which caller is responsible for most of the time spent in MPI: it is the function group Application (and not Forward, Adjoin, cg, or Smoother). If the call tree gets too big to navigate through, use the **Call Graph** to find places in the code that cause expensive calls.



Name	TSelf	TSelf	TTotal	#Cal
All_Processes				
Callers				
Group Application called by idle	1.85521 s		2.3766 s	
Group Application	1.85521 s		2.3766 s	
Callees				
Group Application calling MPI_Comm_size				
Group Application calling MPI_Comm_rank				
Group Application calling MPI_Errhandler_free				
Group Application calling MPI_Errhandler_get				
Group Application calling MPI_Errhandler_set				
Group Application calling MPI_Finalize				
Group Application calling MPI_Bcast				
Group Application calling MPI_Errhandler_create				

Context Menu

Right-click on the chart to open the context menu. The context menu adjusts itself to suit the selected entry in the chart.



Entry	Description
Show All_Processes/xxx in	Show the given profile in a different tab. Here **xxx** stands for the Function group name
Ungroup	Ungroup the selected group and show the distribution of execution time over individual routines or code regions

Regroup	Restore the summarized display after ungrouping. To restore the summarized display after ungrouping a number of times, open the Function Aggregation dialog box (**Advanced > Function Aggregation**) and select **Major Function Groups**.
Function Colors...	Open Function Group Color Editor to apply new colors to function groups
Export (Flat) Data	Save the flat profile data in the text form. In the **Export Text** dialog box that appears, specify the filename or choose the file for data storage. Saving includes all data of the flat profile along with the child processes. The default option is to save them as a .txt file.
Find	Search for a process/function
Chart sub-menu	Print, save, clone and move the Chart (for more details, see section Common Chart Features)
Function Profile Settings	Set or reset the Function Profile preferences
Close Chart	Stop using the Function Profile

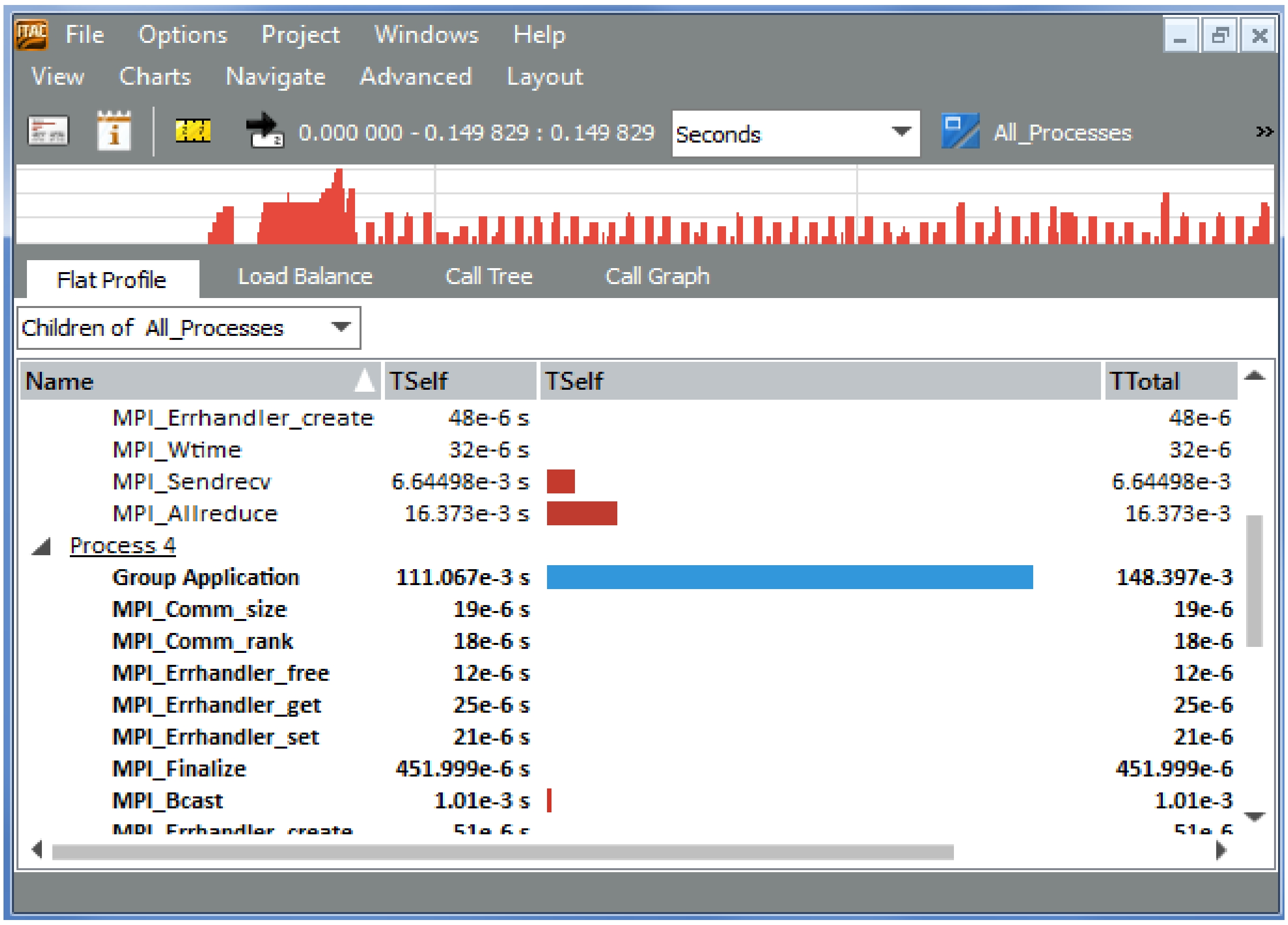
See Also

Common Chart Features

Filtering and Tagging

Tagged entries are shown in bold font. Entries with tagged descendants are shown with underlined names. This helps to see or find the required entry, especially when the tree is large. For more details on tagging and filtering, refer to the Tagging and Filtering section.

Tagged Entries in the Function Profile:



Name	TSelf	TSelf	TTotal
MPI_Errhandler_create	48e-6 s		48e-6
MPI_Wtime	32e-6 s		32e-6
MPI_Sendrecv	6.64498e-3 s		6.64498e-3
MPI_Allreduce	16.373e-3 s		16.373e-3
Process 4			
Group Application	111.067e-3 s		148.397e-3
MPI_Comm_size	19e-6 s		19e-6
MPI_Comm_rank	18e-6 s		18e-6
MPI_Errhandler_free	12e-6 s		12e-6
MPI_Errhandler_get	25e-6 s		25e-6
MPI_Errhandler_set	21e-6 s		21e-6
MPI_Finalize	451.999e-6 s		451.999e-6
MPI_Bcast	1.01e-3 s		1.01e-3
MPI_Errhandler_create	51e-6 s		51e-6

See Also

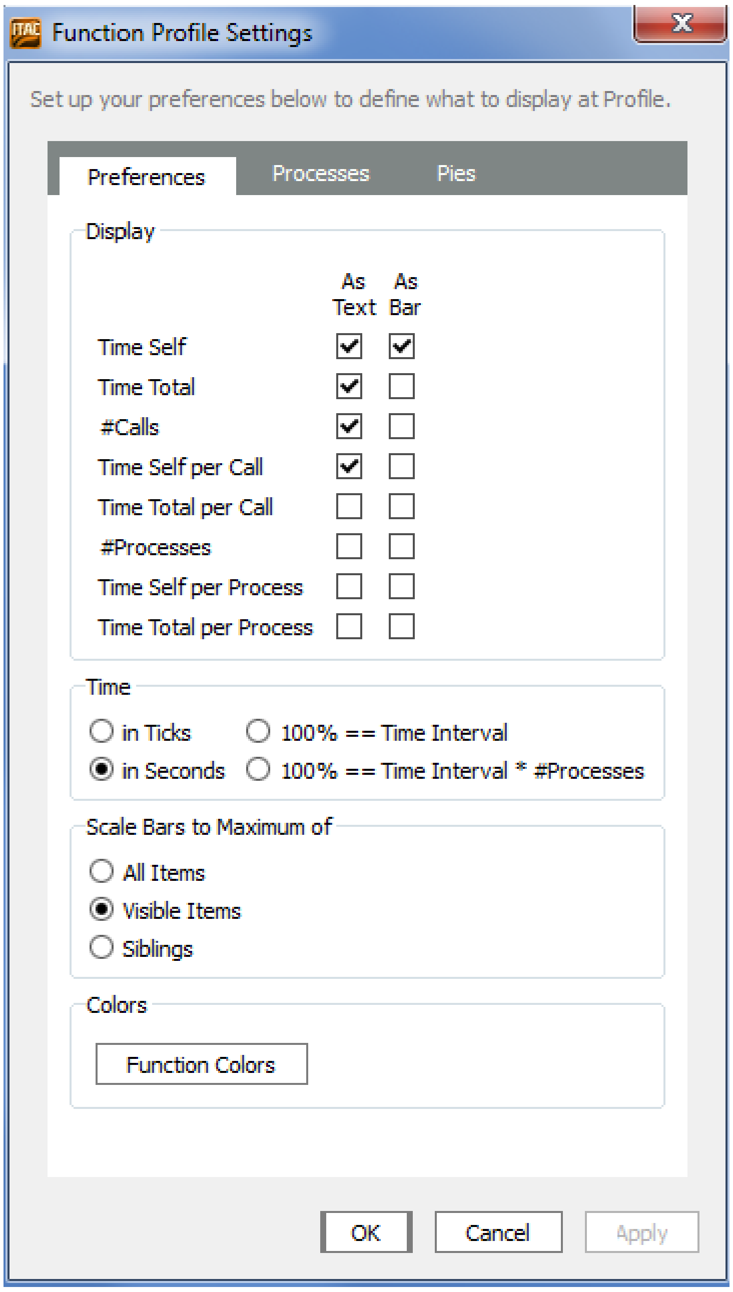
Tagging and Filtering

Function Profile Settings

The **Function Profile Settings** enable you to customize displayed options for all the different views of the Function Profile Chart. To access the Function Profile Settings Preferences, right click on the chart and select **Function Profile Settings** from the context menu.

Preferences

You can set your **Preferences** for the display, time, scale bars and colors used in the Function Profile Chart.



Set up your preferences below to define what to display at Profile.

Display

	As Text	As Bar
Time Self	✓	✓
Time Total	✓	
#Calls	✓	
Time Self per Call	✓	
Time Total per Call		
#Processes		
Time Self per Process		
Time Total per Process		

Time

- in Ticks
- [✓ in Seconds]
- 100% == Time Interval
- 100% == Time Interval * #Processes

Scale Bars to Maximum of

- All Items
- [✓ Visible Items]
- Siblings

Colors

Setting:	**Description:**
Time Self	Display time spent in the given function, excluding time spent in functions called from it.
Time Total	Display time spent in the given function, including time spent in functions called from it
#Calls	Display the number of calls to this function. If other attributes are non-zero, this attribute can be zero, because the actual calls to the respective function can occur outside the current time interval.
Time Self per Call	Display **Time Self** averaged over **#Calls**
Time Total per Call	Display **Time Total** averaged over **#Calls**

#Processes	Display the number of processes in this function
Time Self per Process	Display **Time Self** averaged over **#Processes**
Time Total per Process	Display **Time Total** averaged over **#Processes**
Function Colors	Open the **Function Group Color Editor**

Customize how these attributes are displayed using the adjacent setting for text and a bar graph. By default, the following are displayed - **Time Self**, **Time Total**, **#Calls**, and **Time Self per Call**.

Customize the time format to display as seconds or ticks or as a percentage of the time interval.

Customize the scaling modes are given as radio buttons. They are:

- The default **Visible Items** scales the bars to the respective maximum of all expanded items.
- **All Items** uses the global maximum of all values, regardless of whether they are expanded or not.
- **Siblings** uses only the maximum of the direct siblings.

Processes

If you want particular process to be displayed in the Function Profile:

1. Navigate to the **Processes** tab of the Function Profile Settings and choose the necessary processes
2. Choose the **As selected in Settings** option

To select all but one process:

1. Choose the process you do not need
2. Using the **Invert All** option to reverse the selection

NOTE

It does not influence the current process group of the View, but only focuses the Function Profile on a subset of all processes.

Pies

Toggle between the individual diagram titles and the global legend from the **Pies** control.

See Also

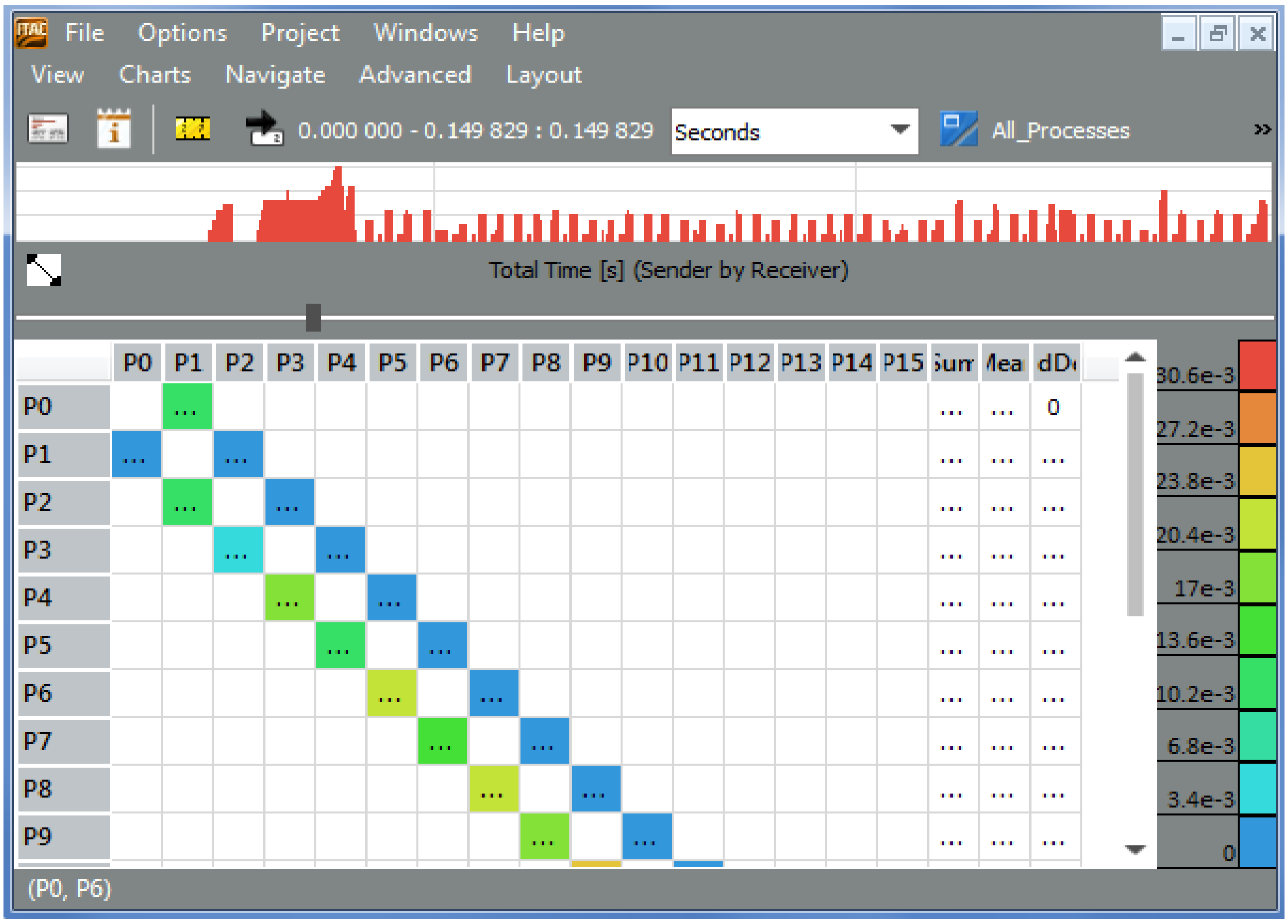
Function Group Color Editor

Message Profile

The Message Profile Chart categorizes messages by groupings in a matrix and shows the value of several attributes in each cell.

By default the matrix is square with the sending processes as row labels and the receiving processes as column labels. It shows in cell (i, j) the total time spent in transferring messages from sender i to receiver j.

This chart also includes per row and per column statistics, which give the sum, the average and the standard deviation for the respective row or column.



Do This:	To Do This:
Click on row or column headers	Select rows or columns
Keep the mouse button pressed and select the respective cells	Select an arbitrary area of the matrix
Right-click and select **Zoom to Selection** from the context menu	Restrict the display to the selected area
Hold down the `[Ctrl]` key and drag the header to the required position	Change the position of the row and column headers
Use the slider above the matrix	Change cell size
Right-click and select the **Message Profile Settings**	Change the grouping that defines the row and column headers of this matrix and therefore the categorization of the data. Available groupings in addition to Sender and Receiver are for example Tag and Communicator.
Use the **Message Profile Settings**	Choose the attribute to be shown in the cells. Apart from the Total Time shown by default there are other time values, transfer rates, volumes, and counts.
Press `[+]` (or `[Ctrl]+[+]`)	Increase the number of digits locally by three (or one) digits

Press `[-]` (or `[Ctrl]+[-]`)	Decrease the number of digits
Place the mouse over any point in the matrix	See the detailed information for the current position in the View status bar in the form `$AttributeValue` (`$RowLabel`, `$ColumnLabel`). It enables you to get exact attribute values even if the cells are configured to be very small or to show no alphanumerical entries at all.

See Also

Message Profile Settings

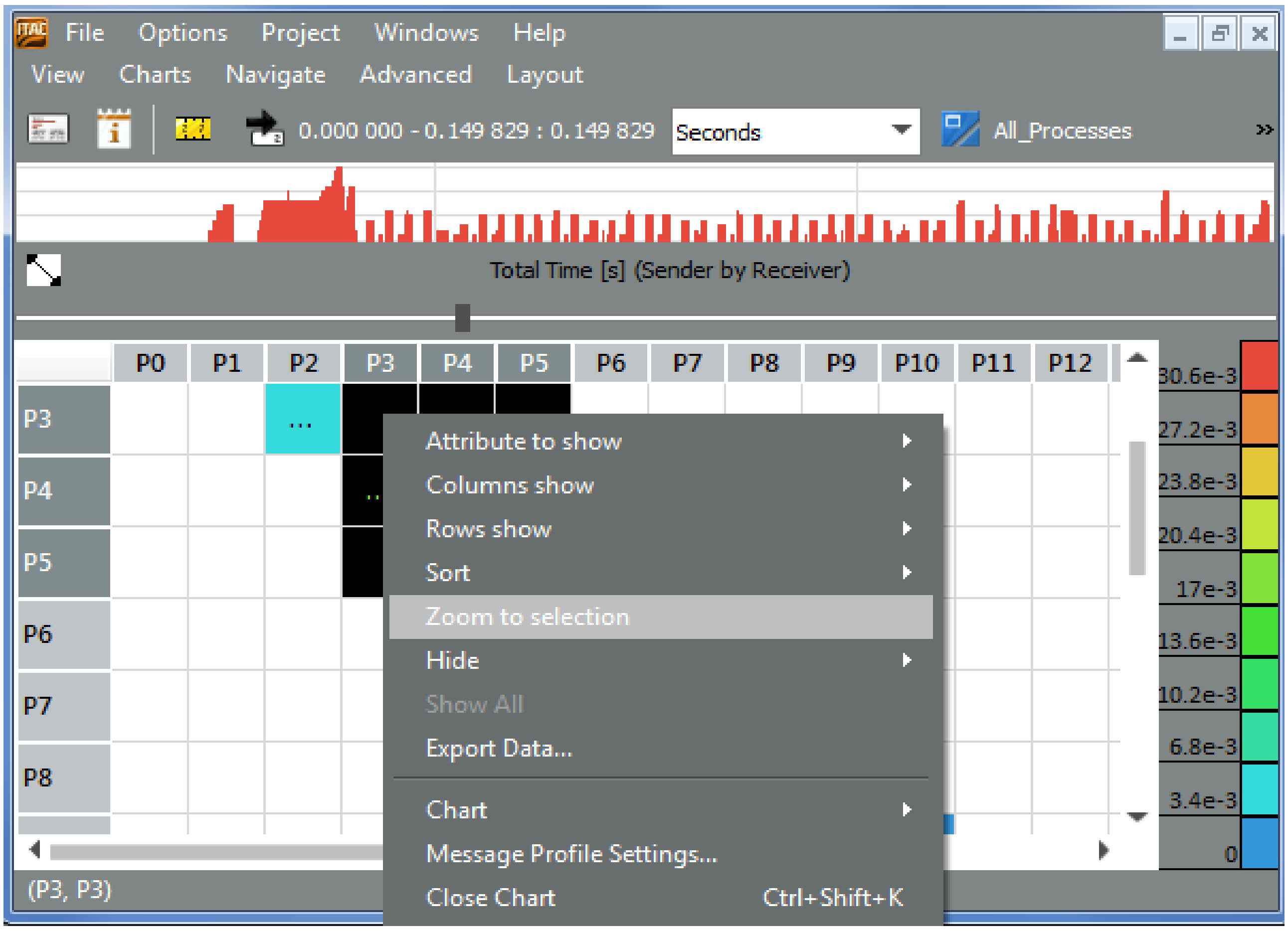
Context Menu

You can access the context menu by right-clicking the Message Profile. Some general context menu options are common to all Charts. Message Profile provides some options that are specific to this timeline.

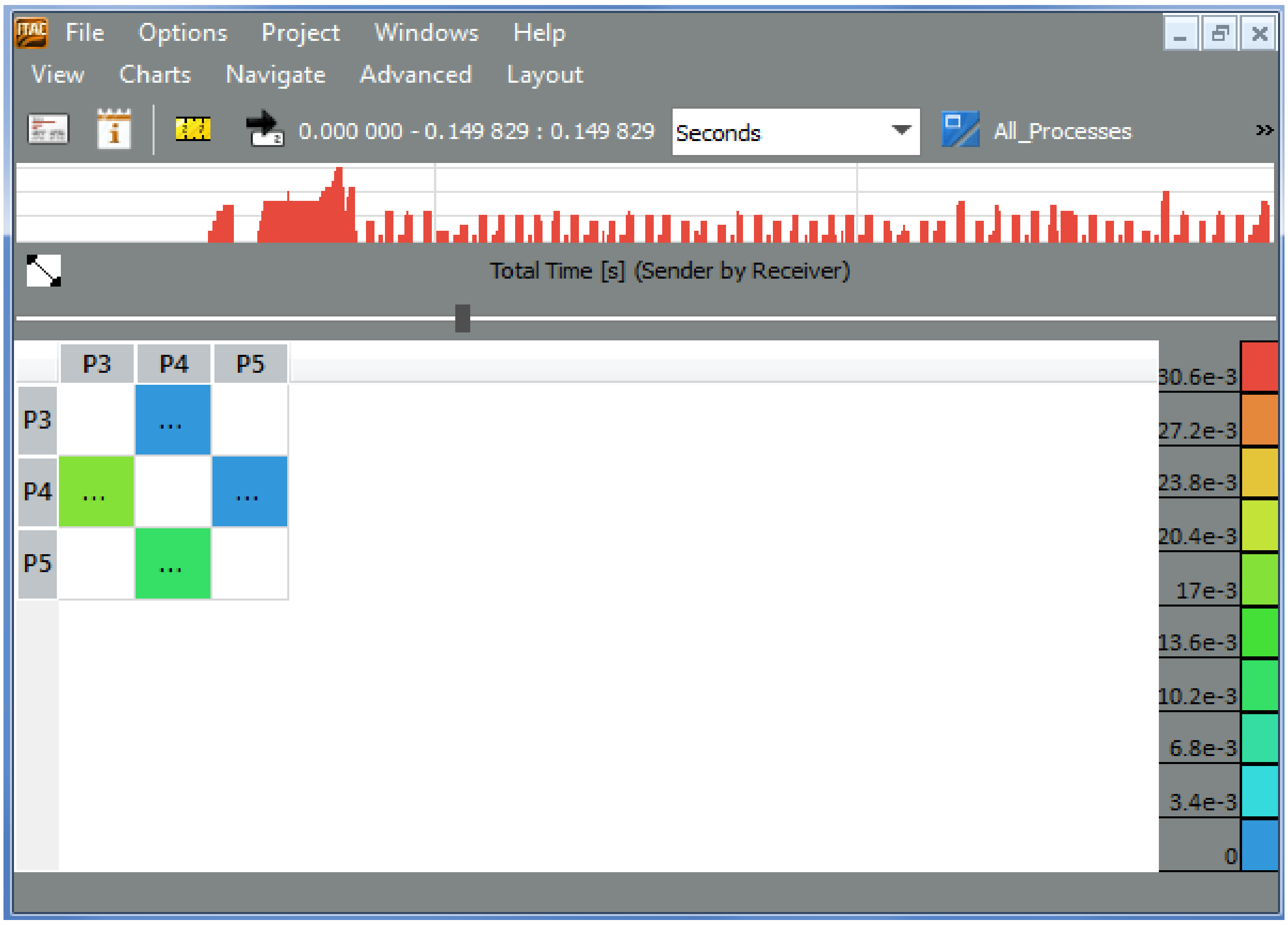
Specific Message Profile context menu options:

Entry:	**Description:**
Attribute to show, **Columns to show** and **Rows to show**	Select attributes and groupings. These entries are the same as those explained in the Message Profile Settings section.
Sort	Sort rows by the values of the column clicked on, or to sort columns by the values in a row clicked on and to switch back to the default order.
Use the **Zoom to the selection** option as shown in the example	See the result. The zoom feature of the Message Profile relies on storing the row and column labels to be suppressed. It can have surprising effects: if **Volume** is selected as row grouping and the rows with labels 17 and 19 are hidden, then scrolling into an area containing messages with volume 18 results in these messages being shown. To suppress all messages with certain volumes, use filtering.
Show All	Show hidden rows and columns
Hide	Hide some data
Export Data	Open an **Export Text** dialog box to select a file to store textual data in. It includes all data cells that contain at least one message, even if they are currently hidden. It does not contain row or column statistics. For each cell, it stores all available attributes.

Zooming to Selected Area in the Message Profile:



Zoomed to the Selected Area:

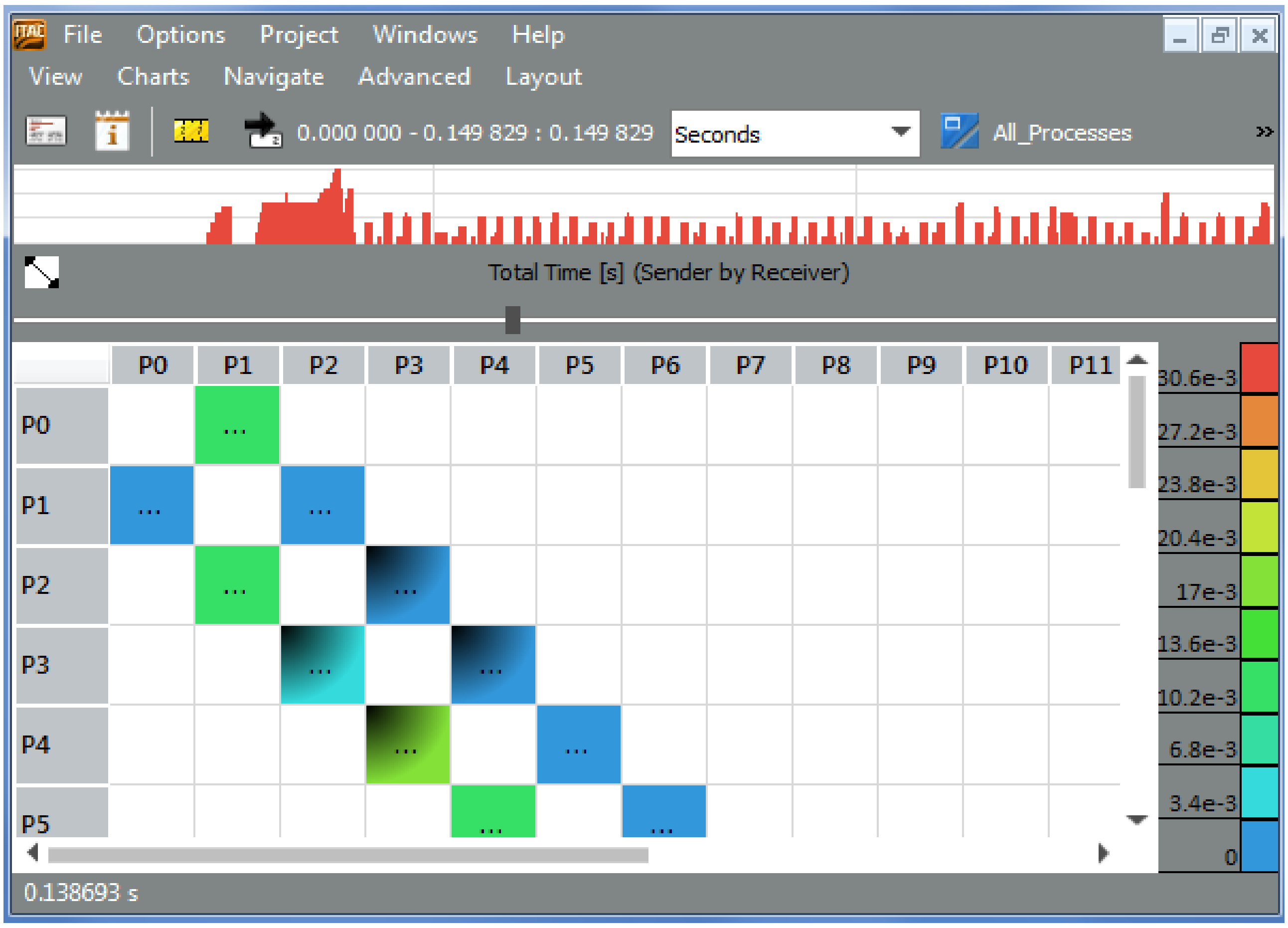


See Also

Message Profile Settings
Common Chart Features
Filtering Dialog Box

Filtering and Tagging

Gradient in the cell color shows that the cell is tagged. A cell is tagged as soon as a single tagged message exists in it. For example, you can see that messages in Process 3 (P3) are tagged:



Messages that do not pass a filter are not accounted for and may result in a smaller matrix when it occurs in empty rows and columns.

See Also

Tagging and Filtering

Aggregation

The View thread group influences the labels of the Sender, Receiver and Sender/Receiver groups. The View function group has no influence.

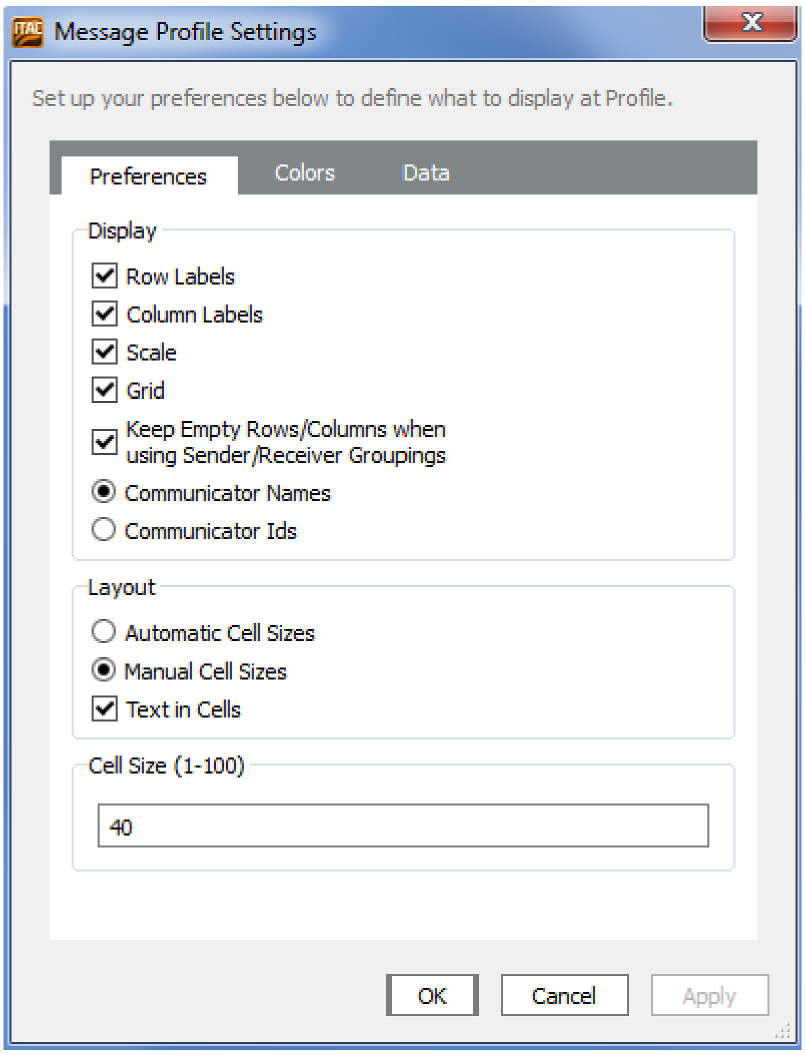
If the View shows the thread group Other, then this results in additional rows and columns for the groupings Sender, Receiver and Sender/Receiver.

Message Profile Settings

The **Settings** dialog box has three subsets: **Preferences**, **Colors** and **Data**.

Preferences

Use the Preferences to configure Message Profile display and layout.


Message Profile Settings

Set up your preferences below to define what to display at Profile.

Display

- [✓] Row Labels
- [✓] Column Labels
- [✓] Scale
- [✓] Grid
- [✓] Keep Empty Rows/Columns when using Sender/Receiver Groupings
- ● Communicator Names
- ○ Communicator Ids

Layout

- ○ Automatic Cell Sizes
- ● Manual Cell Sizes
- [✓] Text in Cells

Cell Size (1-100)

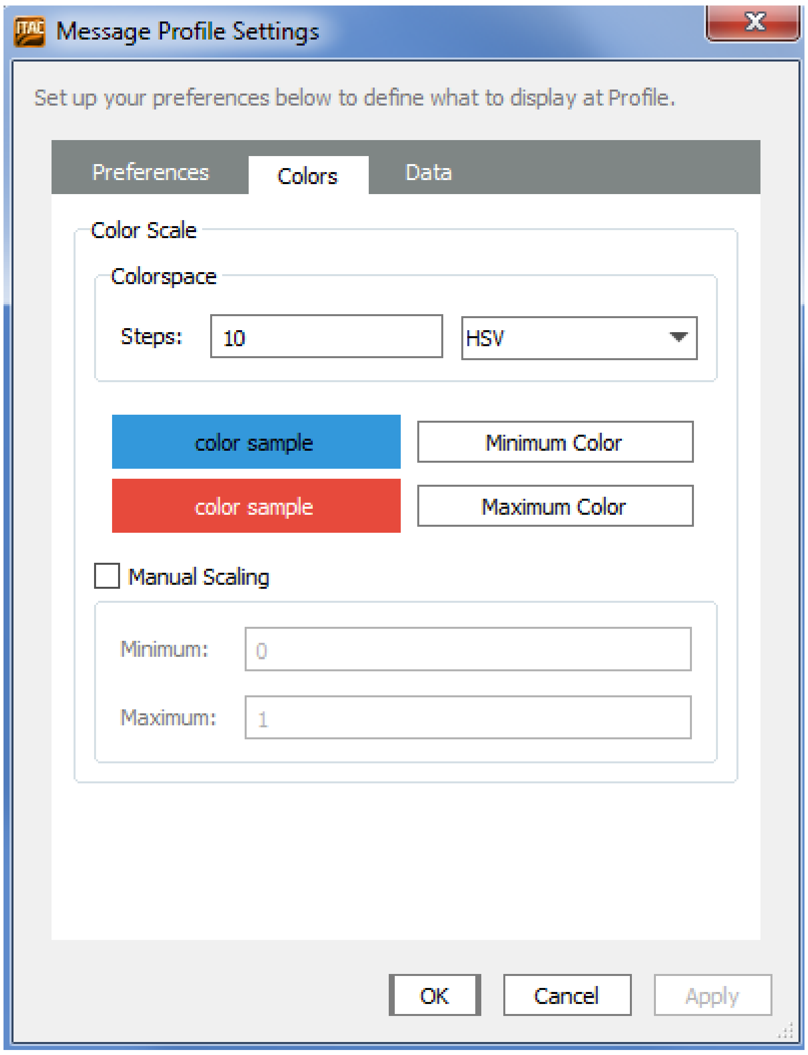
40

Setting	Description
Row Labels	Display row headers
Column Labels	Display column headers
Scale	Display the colored scale next to the matrix
Grid	Control display of black grid lines between cells
Keep Empty Rows/Columns when using Sender/Receiver Groupings	Control display of empty rows and columns. This feature is only relevant for the groupings Sender and Receiver. When enabled, all processes are shown. For example, empty rows and columns will be displayed. It keeps the form of the matrix constant making it easier to see patterns in the data. When disabled, empty rows and columns are not shown in these groupings. All other groupings suppress empty rows and columns to save screen space regardless of the state of this check box.
Communicator Names	Display helpful communicator names (if available in the trace

	file). Displaying communicator names may take a lot of valuable screen space.
Communicator Ids	Display only concise communicator ids
Automatic Cell Sizes	Adjust cell sizes to make all text readable
Manual Cell Sizes	Specify the size of the cells either in the **Cell Size** group at the bottom of the tab. In this mode, the alphanumerical data in the cells is displayed only if it fits or if it is switched off entirely by un-checking the check box **Text in Cells**.

Colors

In this tab you can change the default colors in which the messages in the Message Profile are shown.



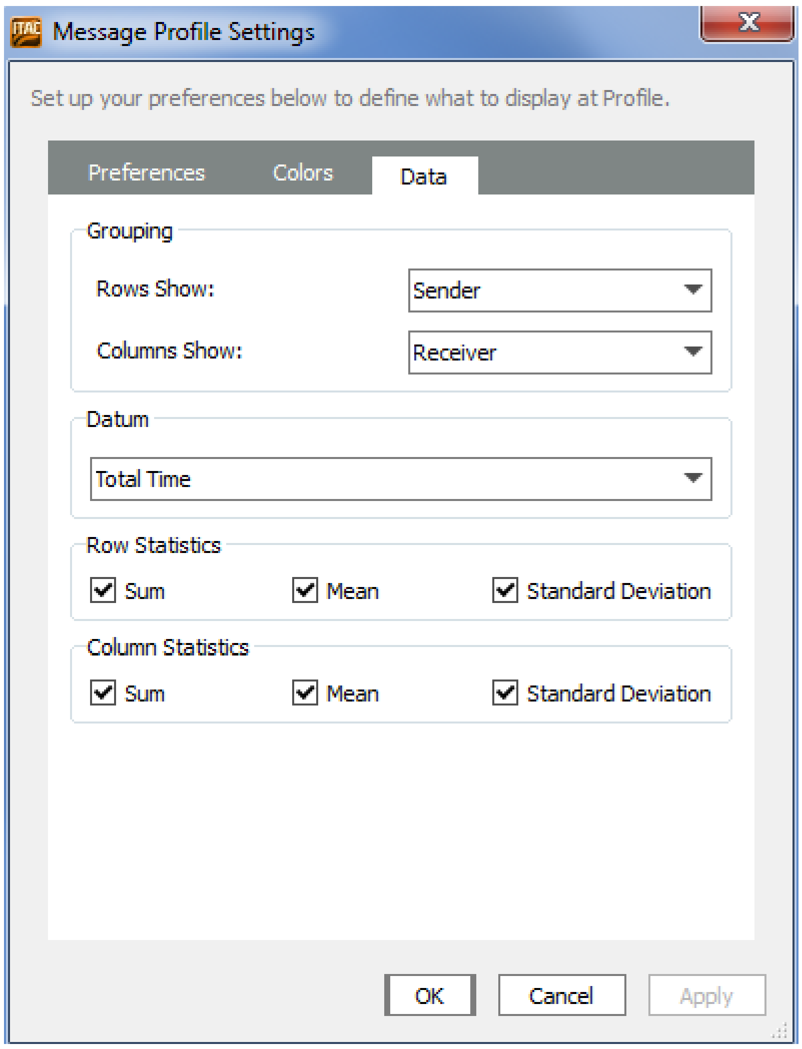
The Message Profile Setting provide the following options:

Setting	**Description**
Steps	Specify the number of color steps (1-255). The chosen colors are considered as points in a color space.

	The scale colors are interpolated on a line through color space connecting these two points.
Maximum Color	Choose the colors for the maximum attribute values
Minimum Color	Choose the colors for the minimum attribute values
HSV/RG	Choose between HSV and RGB color space. HSV is more fancy and colorful, but RGB is often more useful and readable. For monochrome printing, you will get a better result if you choose a very light and a very dark color. For example, white for the minimum and black for the maximum.
Manual Scaling	When enabled you will need to specify the minimum and maximum values for the color scale in the two text input fields below. This is very convenient when you compare two Message Profile Charts that may live in different Views.

Data

The **Grouping** of data is controlled by indicating what appears in rows and columns. Note that not all combinations are possible. For example, you cannot set the same grouping for both rows and columns and you cannot have Sender/Receiver at one axis and one of the Sender or Receiver on the other axis.



Set up your preferences below to define what to display at Profile.

Grouping

Rows Show: Sender

Columns Show: Receiver

Datum

Total Time

Row Statistics

✓ Sum ✓ Mean ✓ Standard Deviation

Column Statistics

✓ Sum ✓ Mean ✓ Standard Deviation

The available groupings are:

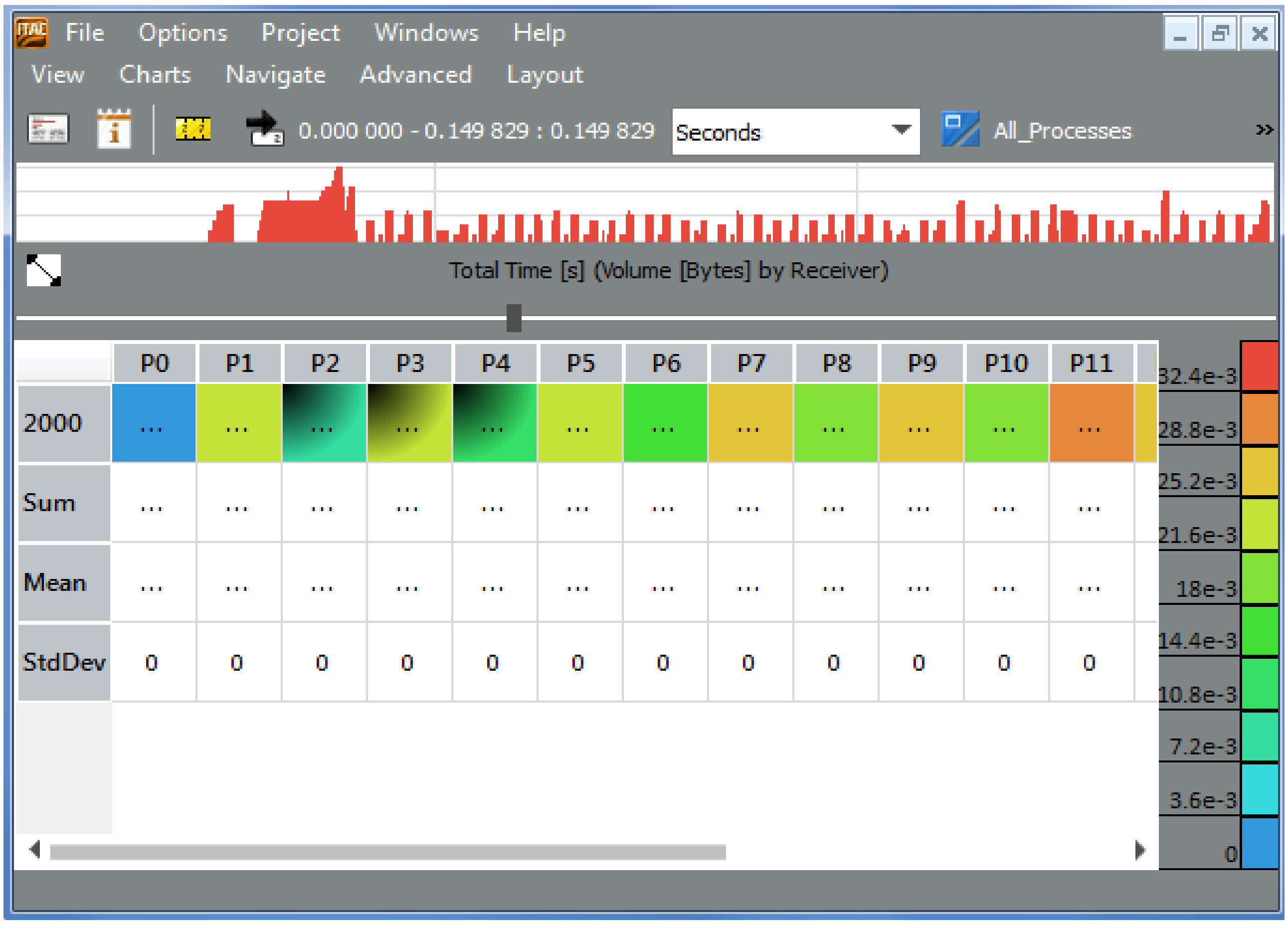
Grouping	**Description**
Sender	Categorizes the messages by Sender. The exact labels are defined by the current thread group that is given by the View (see the Views section).
Receiver	Categorizes the messages by Receiver. The exact labels are defined by the current process group that is given by the View.
Sender/Receiver	Categorizes the messages by Sender/Receiver pairs. The exact labels are defined by the current process group that is given by the View.
Tag	Categorizes the messages by the MPI tag assigned to the message by the program at the sender side.
Communicator	Categorizes the messages by the MPI communicator. The labels are either communicator ids or names. Names are displayed if they are available in the trace file and if they are chosen in the **Preferences** tab of the **Message Profile Settings**

	dialog box.
Volume	Categorizes the messages by their Volume; for example, size in bytes. Grouping Volume by Receiver shows only messages with a volume of 2000 bytes.
Sending Function	Categorizes the messages by the function that sends them. Labels are names of MPI functions such as `MPI_Irsend`. This categorization is not influenced by the current Function Aggregation. This information is only available with traces created by the Intel® Trace Collector version 6 and higher.
Receiving Function	Categorize the messages by the function that receives them. Labels are names of MPI functions like MPI_Waitany. This is not influenced by the current Function Aggregation. This information is only available with traces created by the Intel® Trace Collector version 6 and higher.
Datum group	Allows choosing which attribute should be printed or painted in the cells. See the description of the attributes.
Row Statistics group	Allows switching the individual columns on or off. These columns hold the statistics for the rows.
Column Statistics group	Allows switching the individual rows on or off. These hold the statistics for the columns.

Datum Attributes	**Description**
Total Time	The total travel time of the messages, accumulated over all messages that fall into this cell. The unit is either [s] or [tick] depending on the View setting.
Minimum Time	The minimum travel time of a message, minimized over all messages that fall into this cell. The unit is either [s] or [tick] depending on the View setting.
Maximum Time	The maximum travel time of a message, maximized over all messages that fall into this cell. The unit is either [s] or [tick] depending on the View setting.
Average Transfer Rate, [B/s]	The average transfer rate, averaged over the transfer rates of all messages that fall into this cell. Messages are not weighted; for example, transfer rates of short messages have the same impact as transfer rates of long messages.

Minimum Transfer Rate, [B/s]	The minimum transfer rate, minimized over all messages that fall into this cell.
Maximum Transfer Rate, [B/s]	The maximum transfer rate, maximized over all messages that fall into this cell.
Total Data Volume, [B]	The total data volume, accumulated over all messages that fall into this cell.
Minimum Data Volume, [B]	The minimum data volume, minimized over all messages that fall into this cell.
Maximum Data Volume, [B]	The maximum data volume, maximized over all messages that fall into this cell.
Count, [1]	The number of messages that fall into this cell.

Example of grouping Volume by Receiver:



0.000 000 - 0.149 829 : 0.149 829 Seconds All_Processes

Total Time [s] (Volume [Bytes] by Receiver)

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11
2000	...	...	...	...	...	...	...	...	...	...	...	...
Sum	...	...	...	...	...	...	...	...	...	...	...	...
Mean	...	...	...	...	...	...	...	...	...	...	...	...
StdDev	0	0	0	0	0	0	0	0	0	0	0	0

32.4e-3	🟥
28.8e-3	🟧
25.2e-3	🟨
21.6e-3	🟨
18e-3	🟩
14.4e-3	🟩
10.8e-3	🟩
7.2e-3	🟩
3.6e-3	🟦
0	🟦

See Also

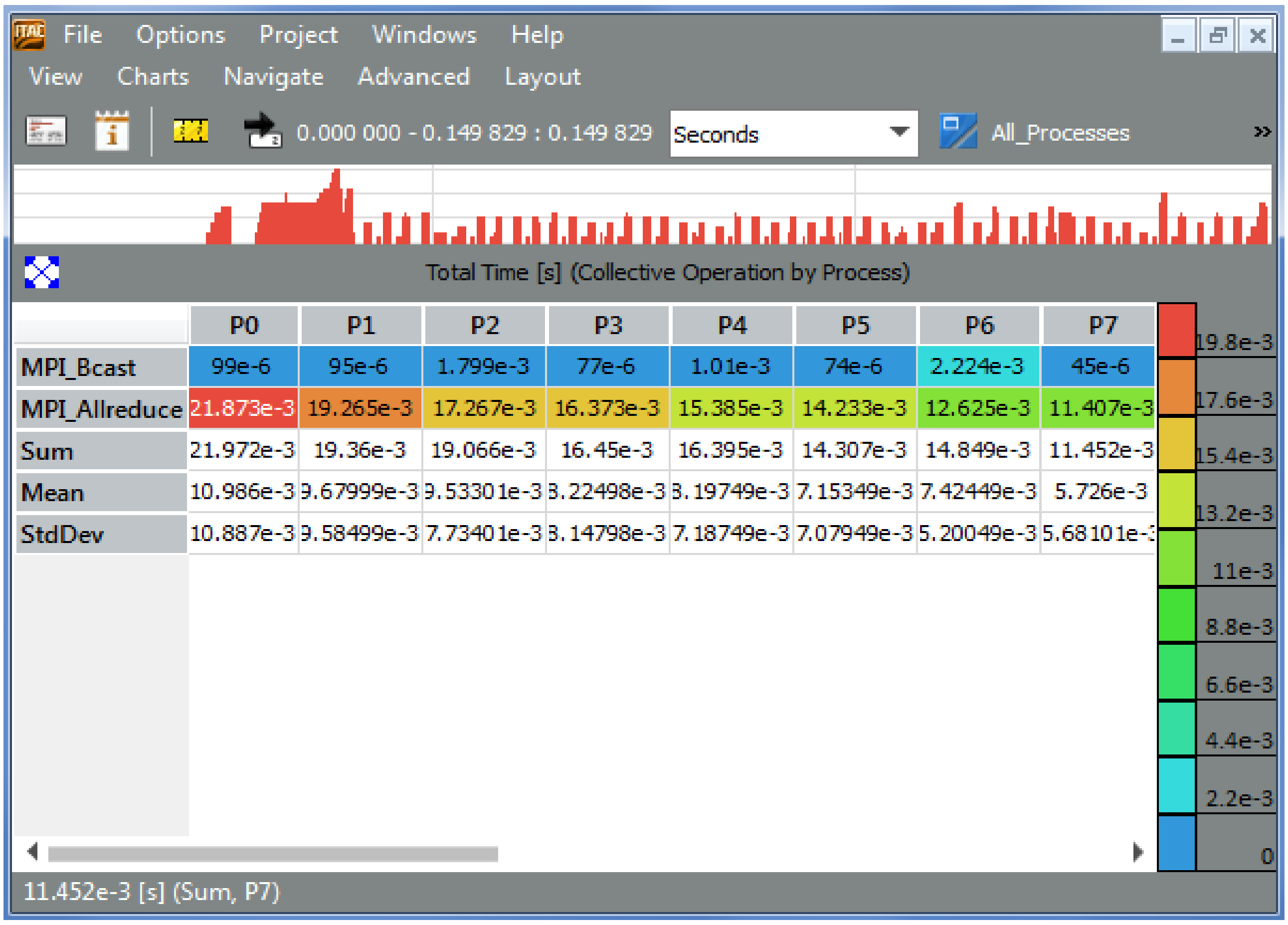
Views

Collective Operations Profile

The Collective Operations Profile enables you to analyze communication patterns that are done using MPI Collective Operations. Like the Message Profile (see Message Profile), the Collective Operations are also represented in a color-coded matrix format. The default matrix shows the type of the Collective Operation as the row label and the process as the column label.

The precision of the values shown can be adjusted as explained in the Message Profile section.

Place the mouse over any point in the matrix and see the detailed information for the current cell in the View status bar in the form `$AttributeValue` (`$RowLabel`, `$ColumnLabel`). It enables you to get exact attribute values even if the cells are configured to be very small or to show no alphanumerical entries at all.



	P0	P1	P2	P3	P4	P5	P6	P7
MPI_Bcast	99e-6	95e-6	1.799e-3	77e-6	1.01e-3	74e-6	2.224e-3	45e-6
MPI_Allreduce	21.873e-3	19.265e-3	17.267e-3	16.373e-3	15.385e-3	14.233e-3	12.625e-3	11.407e-3
Sum	21.972e-3	19.36e-3	19.066e-3	16.45e-3	16.395e-3	14.307e-3	14.849e-3	11.452e-3
Mean	10.986e-3	9.67999e-3	9.53301e-3	8.22498e-3	8.19749e-3	7.15349e-3	7.42449e-3	5.726e-3
StdDev	10.887e-3	9.58499e-3	7.73401e-3	8.14798e-3	7.18749e-3	7.07949e-3	5.20049e-3	5.68101e-

11.452e-3 [s] (Sum, P7)

See Also

Message Profile

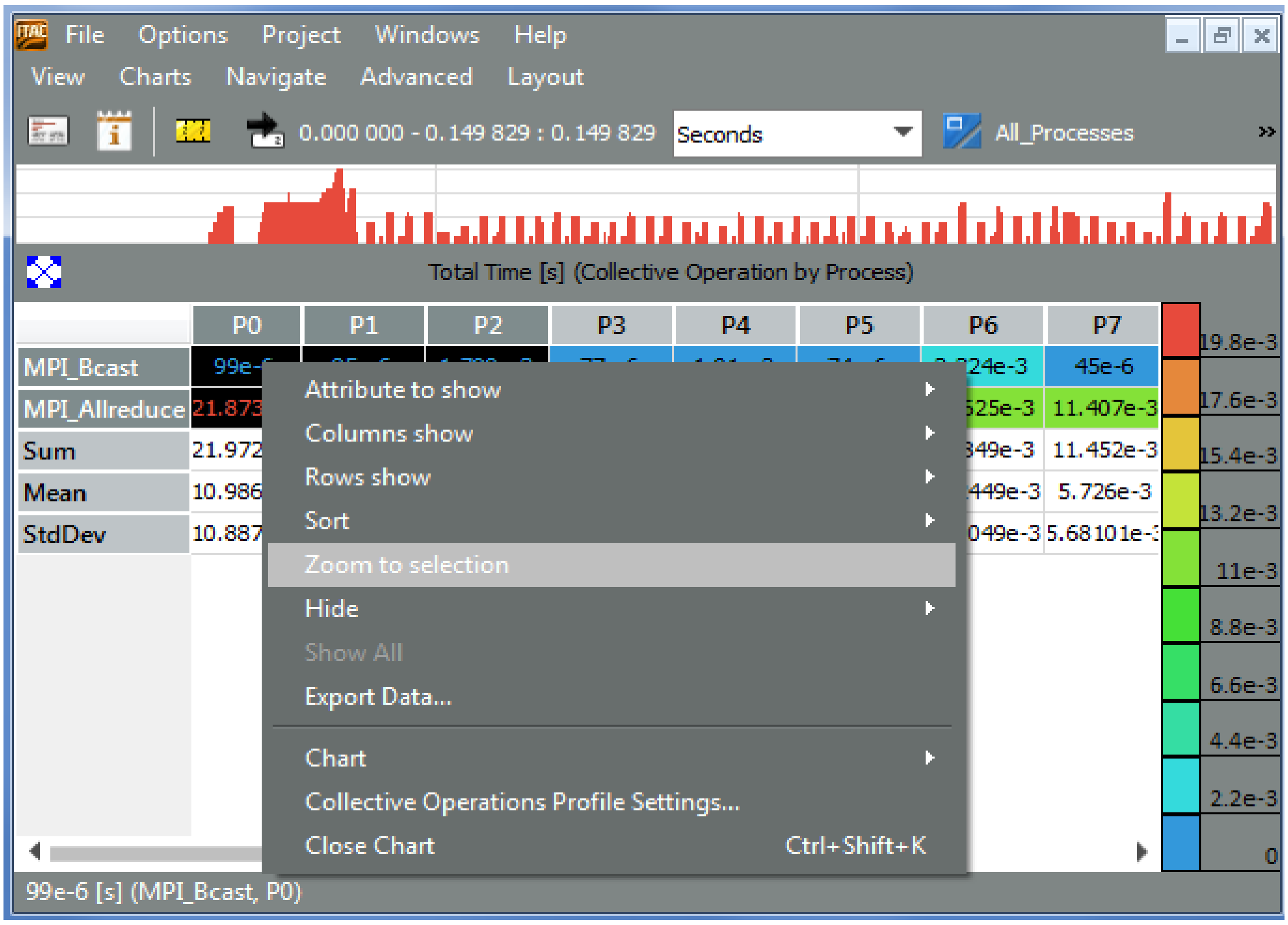
Context Menu

The context menu in the Collective Operations Profile mainly consists of the following entries:

Select This:	**To Do This:**
Attribute to show	Select the attributes to be shown in the Collective Operations Profile
Columns to show	Select if to display Collective Operations Profile by process, by root or by communicator.
Rows to show	Choose whether the rows of the profile show the Collective Operation, the communicator or the root values.
Sort	Sort rows by the values of the column clicked on, or sort columns by the values in a row clicked on and switch back to the default order. It is also useful to switch back to the default order if the

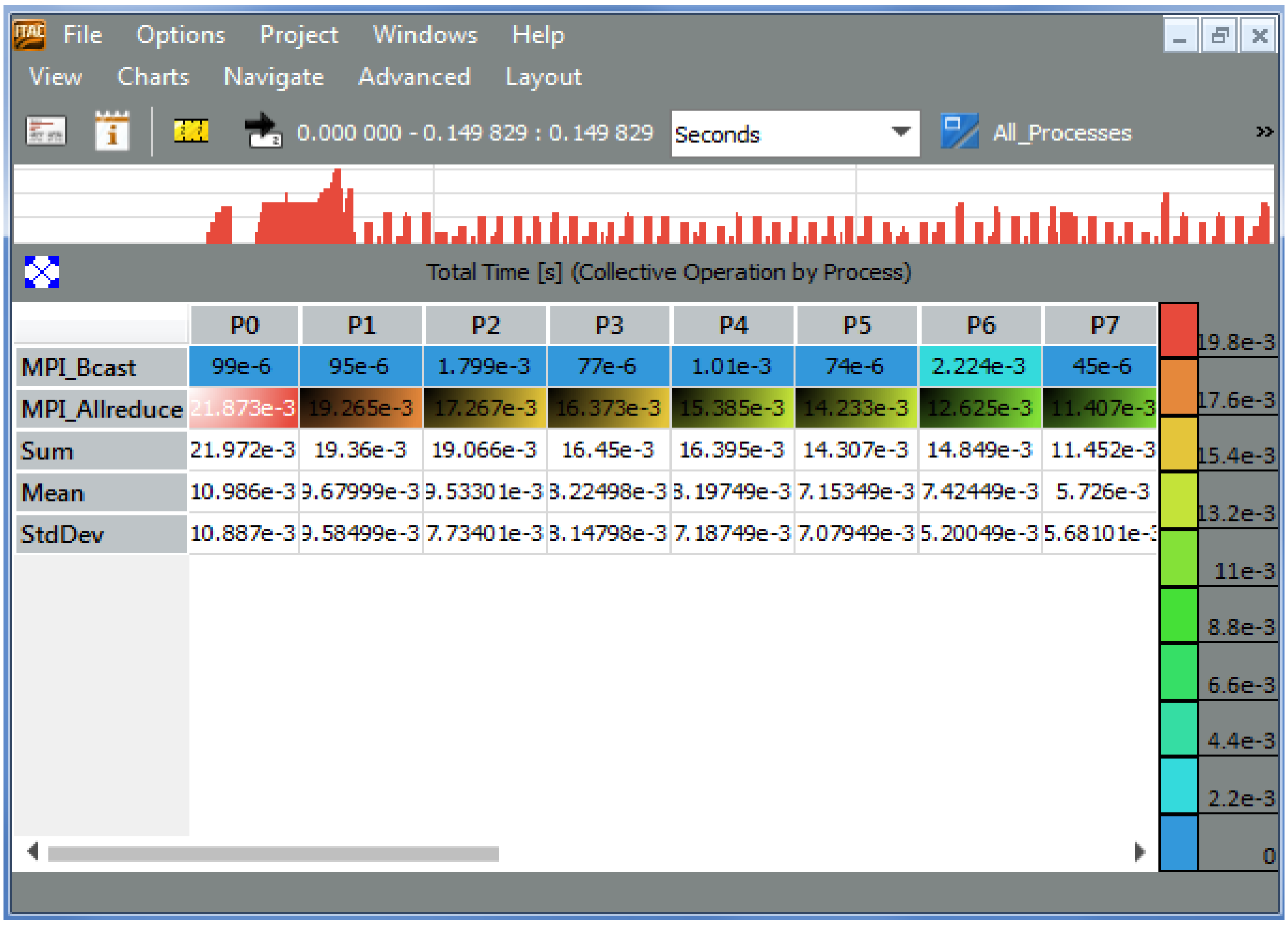
	columns or rows were rearranged by dragging the row or column headers around.
Zoom to selection	Use this entry to focus on a particular region in the matrix. To do this, select the required region with the mouse, right-click the selection and choose the **Zoom to selection** entry from the context menu. See the example. As a result, only the selected region is displayed.
Hide	Hide all selected cells. To select the necessary cells, hold down the left mouse button and move over the required region. It also automatically opens the context menu.
Show All	Display all the hidden cells. It is enabled only if cells have been previously hidden using the **Hide** entry.
Export Data	Open an **Export Text** dialog box and select a file to store textual data in. This includes all data cells that contain at least one message, even if they are currently hidden. For each cell, all available attributes are given. It does not contain row or column statistics.
Collective Operations Profile Settings	Open the **Collective Operations Profile Settings** dialog box

Zooming to Selection in Collective Operations Profile:



Filtering and Tagging

Tagged cells are emphasized by a small additional frame around the cell in the color of the alphanumerical entry in the cell. A cell is tagged as soon as a single tagged message falls into that cell. Here is an example of tagging `MPI_Allreduce` function in Collective Operations Profile:



Total Time [s] (Collective Operation by Process)

	P0	P1	P2	P3	P4	P5	P6	P7
MPI_Bcast	99e-6	95e-6	1.799e-3	77e-6	1.01e-3	74e-6	2.224e-3	45e-6
MPI_Allreduce	21.873e-3	19.265e-3	17.267e-3	16.373e-3	15.385e-3	14.233e-3	12.625e-3	11.407e-3
Sum	21.972e-3	19.36e-3	19.066e-3	16.45e-3	16.395e-3	14.307e-3	14.849e-3	11.452e-3
Mean	10.986e-3	9.67999e-3	9.53301e-3	8.22498e-3	8.19749e-3	7.15349e-3	7.42449e-3	5.726e-3
StdDev	10.887e-3	9.58499e-3	7.73401e-3	8.14798e-3	7.18749e-3	7.07949e-3	5.20049e-3	5.68101e-

Filtering in the Collective Operations Profile works the same way as in any other Chart: only the chosen Collective Operations are shown, the rest are filtered out.

See Also

Tagging and Filtering

Collective Operations Profile Settings

Use the **Collective Operations Settings** to change the settings of the Collective Operations Profile and the Chart display. You can change the colors, the layout and the statistical attributes.

The **Collective Operation Profile Settings** dialog box is divided into three tabs namely the **Preferences** tab, the **Colors** tab and the **Data** tab.

Preferences Tab

Use the **Preferences** tab to adjust the **Display** settings and the **Layout** settings.

Setting	Description
Row Labels	Control the display of the row headers
Column Labels	Control the display of the column headers
Scale	Display the colored scale that is seen on the right-hand side of the matrix
Grid	Control display of the black grid around the cells
Keep Empty Rows/Columns when using Sender/Receiver	Switches a special feature that is only relevant for the Groupings Sender and Receiver.

Groupings	For these groupings, when enabled all processes should always be shown, for example, include displaying empty rows and columns. It keeps the form of the matrix constant and makes it easier to identify patterns. When disabled, empty rows and columns even for these groupings are suppressed. All other groupings suppress empty rows and columns to save screen space regardless of the state of this check box.
Communicator Names	Display helpful communicator names (if available in the trace file). This may take a lot of valuable screen space
Communicator Ids	Restrict the display to show only concise communicator ids.
Automatic Cell Sizes mode, checking **Equal Cell Sizes**	Make each column equal in width. When selected, **Square Cell Sizes** attribute is enabled
Square Cell Sizes	Render square cells.
Manual Cell Sizes	Specify the size of the cells. This is done either in the **Cell Size** group at the bottom of the tab or by sizing the cells manually with the slider that is available on top of the matrix as soon as this setting is applied. In this mode, the alphanumerical data in the cells is displayed only if it fits. Otherwise, it is switched off entirely by unchecking the check box **Text in Cells**.

Colors

Setting	Description
Maximum Color	Choose the colors for the maximum attribute values
Minimum Color	Choose the colors for the minimum attribute values
Steps	Specify the number of color steps (1-255)

The chosen colors are treated as points in a color space and the colors of the scale are interpolated on a line through color space connecting these two points. You can select space to use either HSV or RGB. HSV is fancier and colorful, but RGB is often more useful and readable. For monochrome printing, it is advisable to choose a very light and a very dark color. You can choose white for the minimum and black for the maximum.

By enabling **Manual Scaling** it is possible to specify the minimum and maximum values for the color scale in the following two fields.

Data

Customize how the Collective Operations data is displayed and analyzed. The **Data** features are subdivided into the **Grouping** section, the **Datum** section, the **Row Statistics** and the **Column Statistics**.

Setting	Description
Grouping	Choose how the data is grouped into categories by independently selecting what is in the rows and columns.S Note that not all combinations are possible. For example, you cannot have the Communicator for both the row and column.

	All the header values are explained below.
Communicator	Categorize the messages by the MPI communicator. The labels are either communicator ids or names. Names are displayed if they are available in the trace file and if they are chosen in the **Preferences** tab of the **Settings** dialog box.
Collective Operation	Display the types of operations like MPI_Allreduce and MPI_Bcast.
Root	Display the root used in the operation, if applicable. If there is no root, a label No Root is created.
Process	Categorize the operations by the processes.
Datum	Choose which attribute should be printed or painted in the cells.
Total Time	The total time spent in operations, accumulated over all operations and all processes referred in this cell. For a single process and a single operation this is the time spent in the call to the operation. For cells referring to a process group this is the sum of the times all contained processes did spent in the operation. For many operations it is the sum over the times spent in each single operation. The unit can either be seconds [s] or ticks [tick] depending on the View setting.
Minimum Time	The minimum time spent in an operation, minimized over all operations and all processes that fall into this cell ([s] or [tick])
Maximum Time	The maximum time spent in an operation, maximized over all operations and all processes that fall into this cell ([s] or [tick])
Total Volume Sent	The total data volume that has been sent from all operations in this cell [bytes]
Minimum Volume Sent	The minimum amount of data volume that has been sent by an operation in this cell [bytes]
Maximum Volume Sent	The maximum amount of data volume that has been sent by an operation in this cell [bytes]
Total Volume Received	The total data volume that has been received by all operations in this cell [bytes]
Minimum Volume Received	The minimum amount of data volume that has been sent by an operation in this cell [bytes]
Maximum Volume Received	The maximum amount of data volume that has been received by an operation in this cell [bytes]
Total Data Volume	The total data volume, accumulated over all operations in this cell

Count	The number of operations in this cell
Row Statistics	Specify whether the statistical values like the sum, the mean or the standard deviation should be displayed for the rows. Similarly, Column Statistics give the above mentioned statistical values for the given columns.

Performance Assistant

Performance Assistant provides you with the general and detailed information about performance problems. You can use this feature either in GUI or in the CLI mode.

To access the tool from the GUI, go to **Charts** > **Performance Assistant** or use the Σ toolbar button. For information on how to use the Performance Assistant in the CLI, refer to the Command Line Interface (CLI) section.

By default, the Performance Assistant does not display all performance problems in your application so as not to slow down the work of the Intel® Trace Analyzer. If you want to get information on all the performance issues, select **Show advanced...** . Note that this option resets zoom to default. It can also slow down the work of the Intel Trace Analyzer.

NOTE

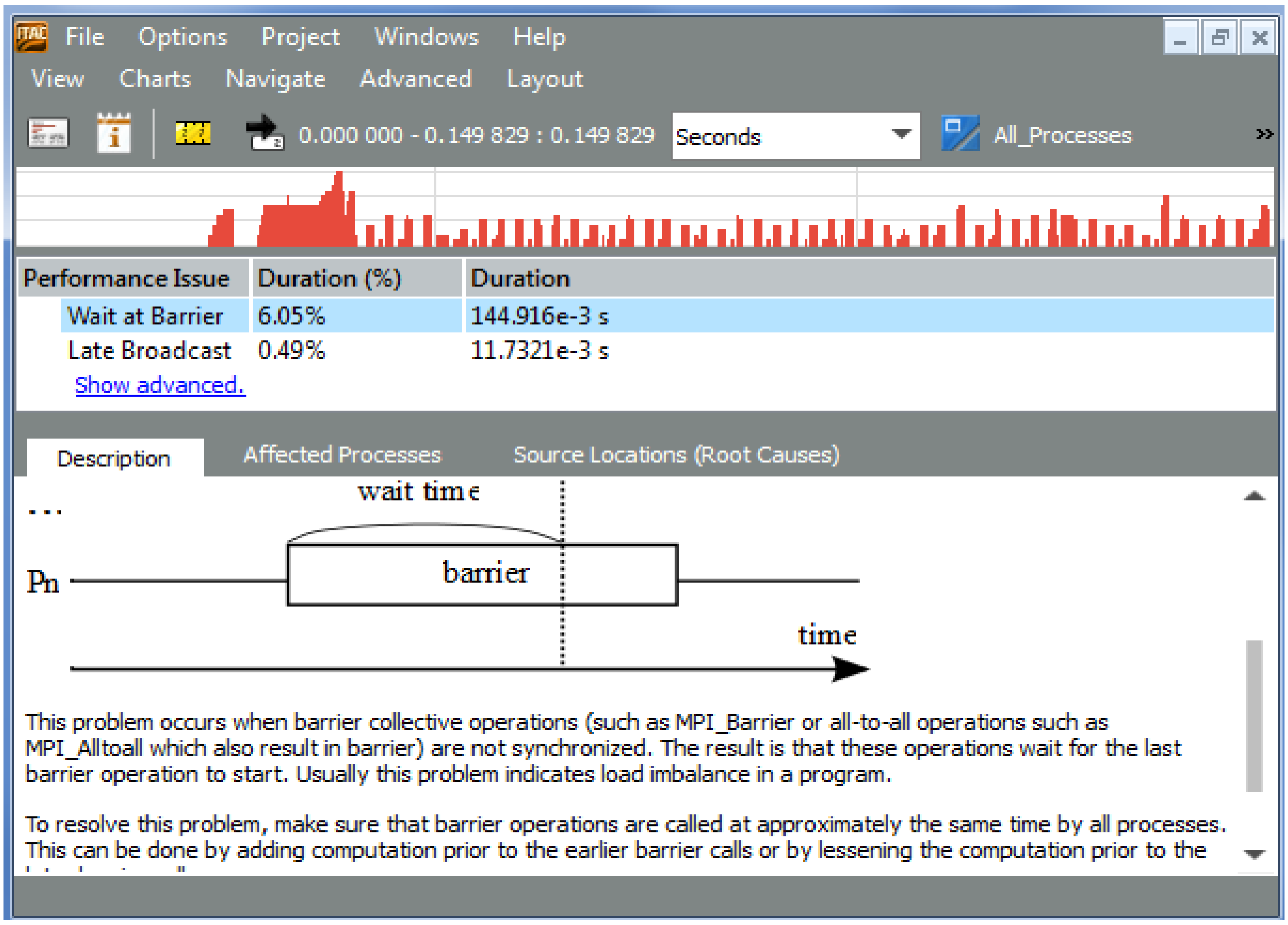
Performance Assistant is not supported when the trace file is collected with enabled Correctness Checker or `TIME-WINDOWS`. In this case, the Performance Assistant displays incorrect data.

See all discovered performance problems and their duration in the upper section of the Performance Assistant. The duration of the problem is displayed in two values - the absolute value and the percentage of the application running time.

Use the lower section of the Performance Assistant to get detailed information on the selected problem. The lower section contains the following tabs:

Description

In the Description tab, get the description of the problem and recommendations on how to fix it. For example:



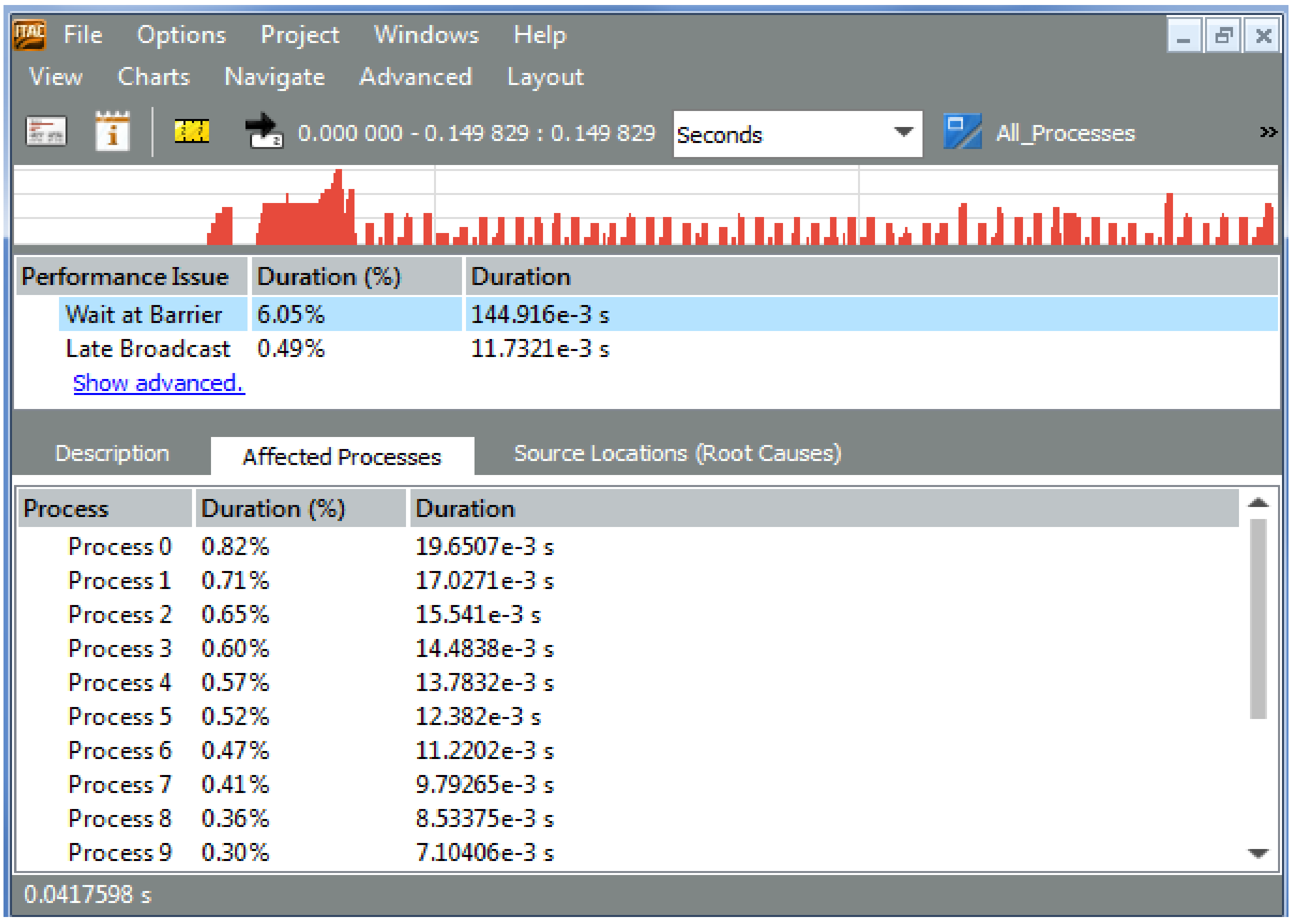
Performance Issue	Duration (%)	Duration
Wait at Barrier	6.05%	144.916e-3 s
Late Broadcast	0.49%	11.7321e-3 s
Show advanced.		

This problem occurs when barrier collective operations (such as MPI_Barrier or all-to-all operations such as MPI_Alltoall which also result in barrier) are not synchronized. The result is that these operations wait for the last barrier operation to start. Usually this problem indicates load imbalance in a program.

To resolve this problem, make sure that barrier operations are called at approximately the same time by all processes. This can be done by adding computation prior to the earlier barrier calls or by lessening the computation prior to the

Affected Processes

In the Affected Processes tab, see the distribution of the performance problem duration among the processes. For example:



Performance Issue	Duration (%)	Duration
Wait at Barrier	6.05%	144.916e-3 s
Late Broadcast	0.49%	11.7321e-3 s

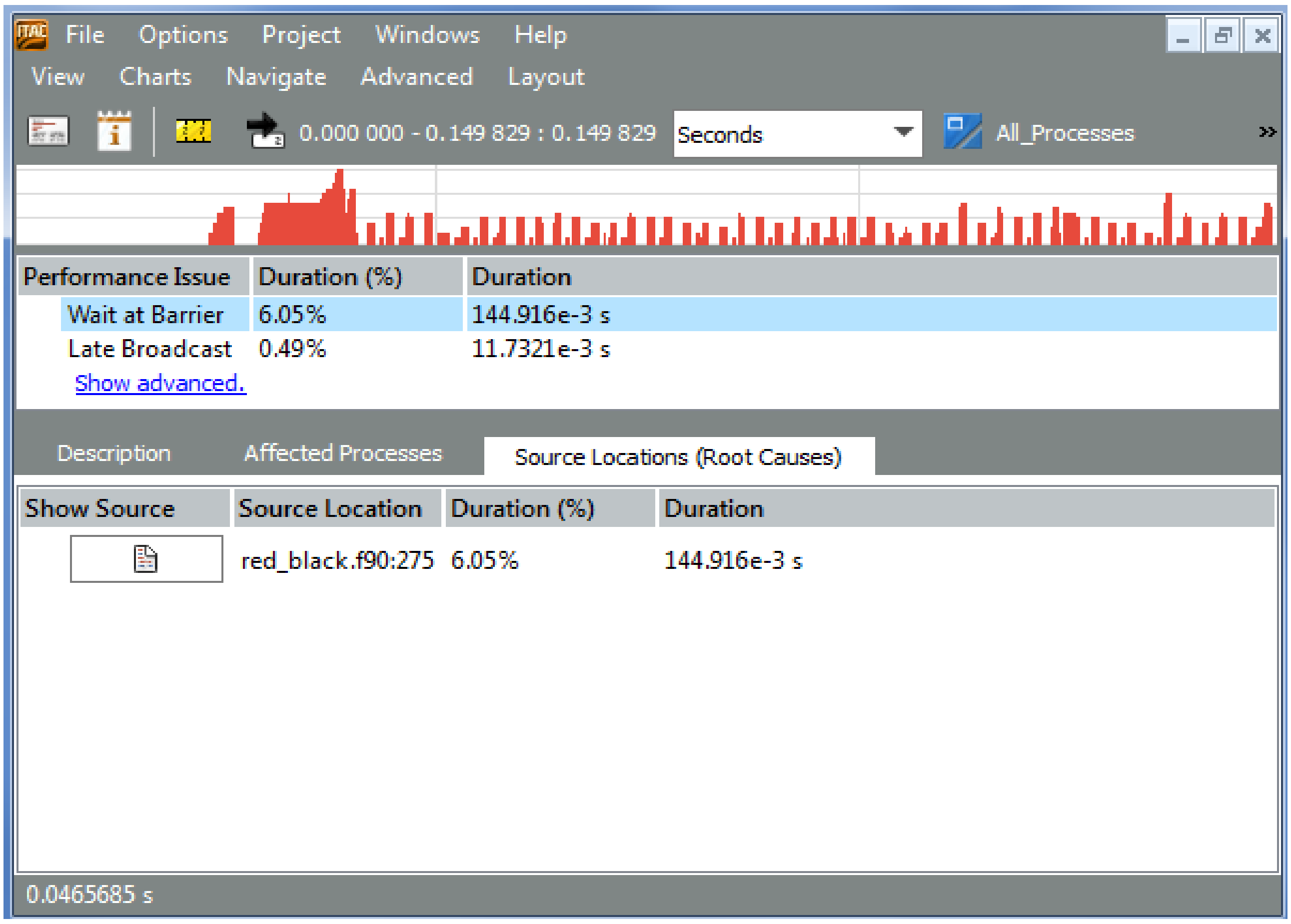
Process	Duration (%)	Duration
Process 0	0.82%	19.6507e-3 s
Process 1	0.71%	17.0271e-3 s
Process 2	0.65%	15.541e-3 s
Process 3	0.60%	14.4838e-3 s
Process 4	0.57%	13.7832e-3 s
Process 5	0.52%	12.382e-3 s
Process 6	0.47%	11.2202e-3 s
Process 7	0.41%	9.79265e-3 s
Process 8	0.36%	8.53375e-3 s
Process 9	0.30%	7.10406e-3 s

Source Locations (Root Causes)

Go to the Source Locations (Root Causes) tab, to get instances of the problem in the source code.

NOTE

Performance Assistant displays source code only if debug information is present in it and the VT_PCTRACE environment variable is enabled.

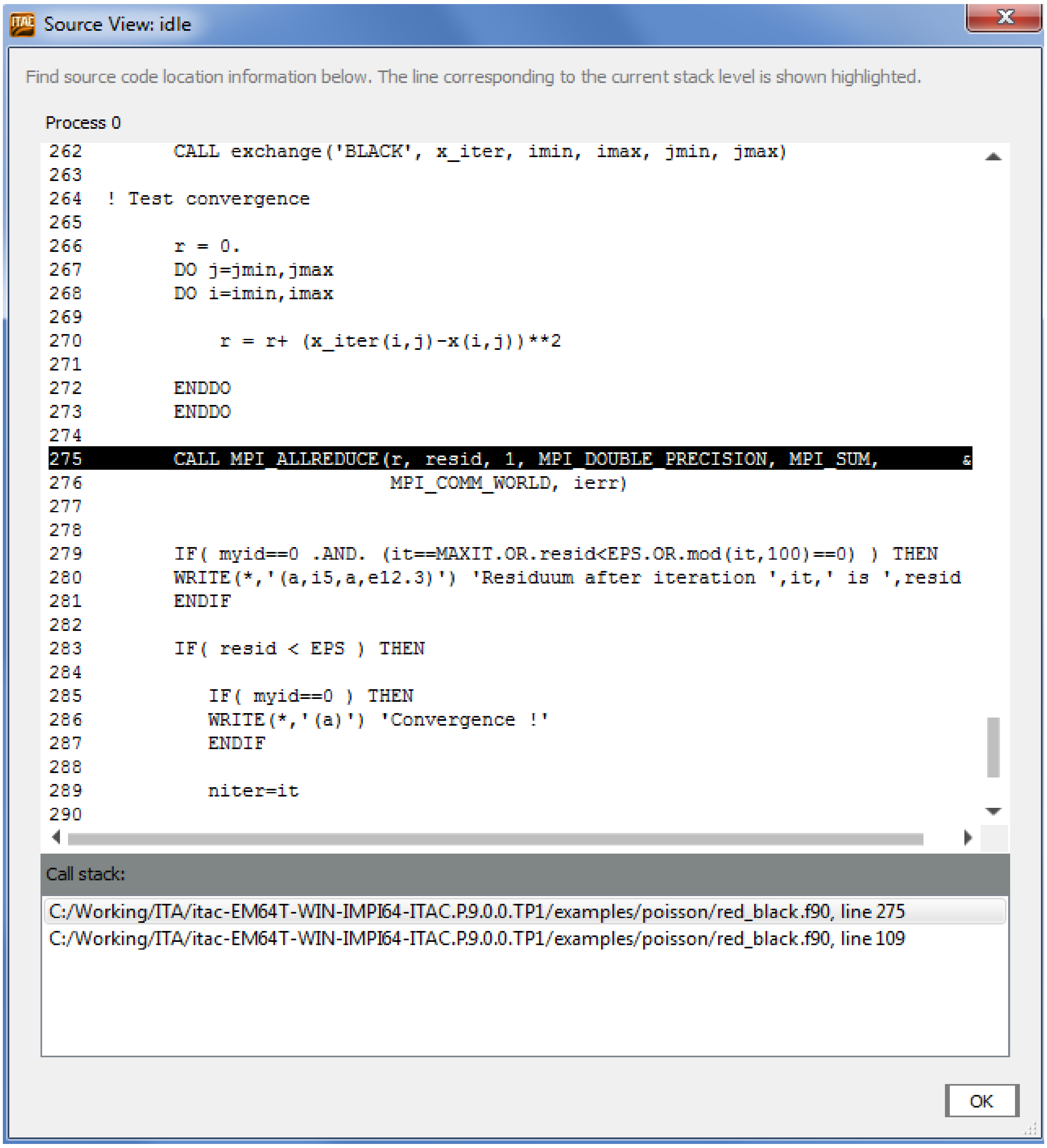


Performance Issue	Duration (%)	Duration
Wait at Barrier	6.05%	144.916e-3 s
Late Broadcast	0.49%	11.7321e-3 s

Show Source	Source Location	Duration (%)	Duration
🗎	red_black.f90:275	6.05%	144.916e-3 s

Press the **Show Source** button to open the Source View dialog box and see the problem-causing calls. Change the highlighted code to fix the problem (get the possible solutions under the Description tab of the Performance Assistant). The Source View dialog box also contains the call stack information.

For example:



Find source code location information below. The line corresponding to the current stack level is shown highlighted.

Process 0

```
      CALL exchange('BLACK', x_iter, imin, imax, jmin, jmax)

! Test convergence

      r = 0.
      DO j=jmin,jmax
      DO i=imin,imax

         r = r+ (x_iter(i,j)-x(i,j))**2

      ENDDO
      ENDDO

      CALL MPI_ALLREDUCE(r, resid, 1, MPI_DOUBLE_PRECISION, MPI_SUM,        &
                         MPI_COMM_WORLD, ierr)


      IF( myid==0 .AND. (it==MAXIT.OR.resid<EPS.OR.mod(it,100)==0) ) THEN
      WRITE(*,'(a,i5,a,e12.3)') 'Residuum after iteration ',it,' is ',resid
      ENDIF

      IF( resid < EPS ) THEN

         IF( myid==0 ) THEN
         WRITE(*,'(a)') 'Convergence !'
         ENDIF

         niter=it
```

Call stack:

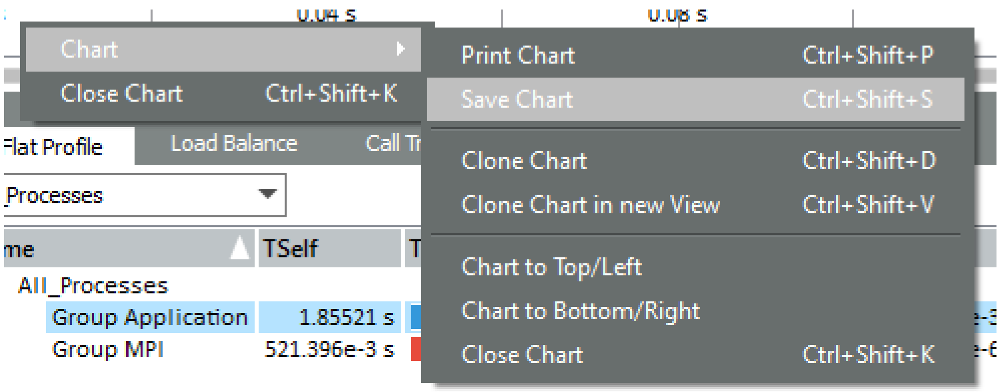
C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9.0.0.TP1/examples/poisson/red_black.f90, line 275
C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9.0.0.TP1/examples/poisson/red_black.f90, line 109

See Also

Command Line Interface (CLI)
Source View Dialog Box

Common Chart Features

All Charts share some common features that are available through common context menu entries.



Select This:	**Shortcut:**	**To Do This:**
Print Chart	`Ctrl+Shift+P`	Print a hard copy of the Chart
Save Chart	`Ctrl+Shift+S`	Save the chart as a picture
Clone Chart	`Ctrl+Shift+D`	Open exactly the same Chart within the present View. In other words, clone the current Chart.
Clone Chart in New View	`Ctrl+Shift+V`	Create a new View and open a copy of the current Chart in the new View.
Chart to Top/Left	None	Useful when two or more Charts of the same kind (Timelines or Profiles) are open. If the Charts are placed one on top of each other, use this entry to move the selected Chart to the top. If they are placed next to each other, then choose this option to move the select Chart to the left.
Chart to Bottom/Right	None	Useful when two or more Charts of the same kind (Timelines or Profiles) are open. If the Charts are placed one on top of each other, use this entry to move the selected Chart to the bottom. If they are placed next to each other, choose this option to move the selected Chart to the right.
Close Chart	`Ctrl+Shift+K`	Close the Chart
Ungroup Function Group	None	Ungroup the selected function group
Regroup Function Group	None	Restore the summarized display after ungrouping

2.2.7. Dialogs

Apart from the Settings dialog boxes of each chart, there are a number of other dialog boxes in the Intel® Trace Analyzer. All dialog boxes have the same semantics regarding the buttons **OK**, **Cancel** and **Apply**.

In case the current settings of the dialog boxes are inconsistent or out of bounds, the **OK** and **Apply** buttons are both disabled.

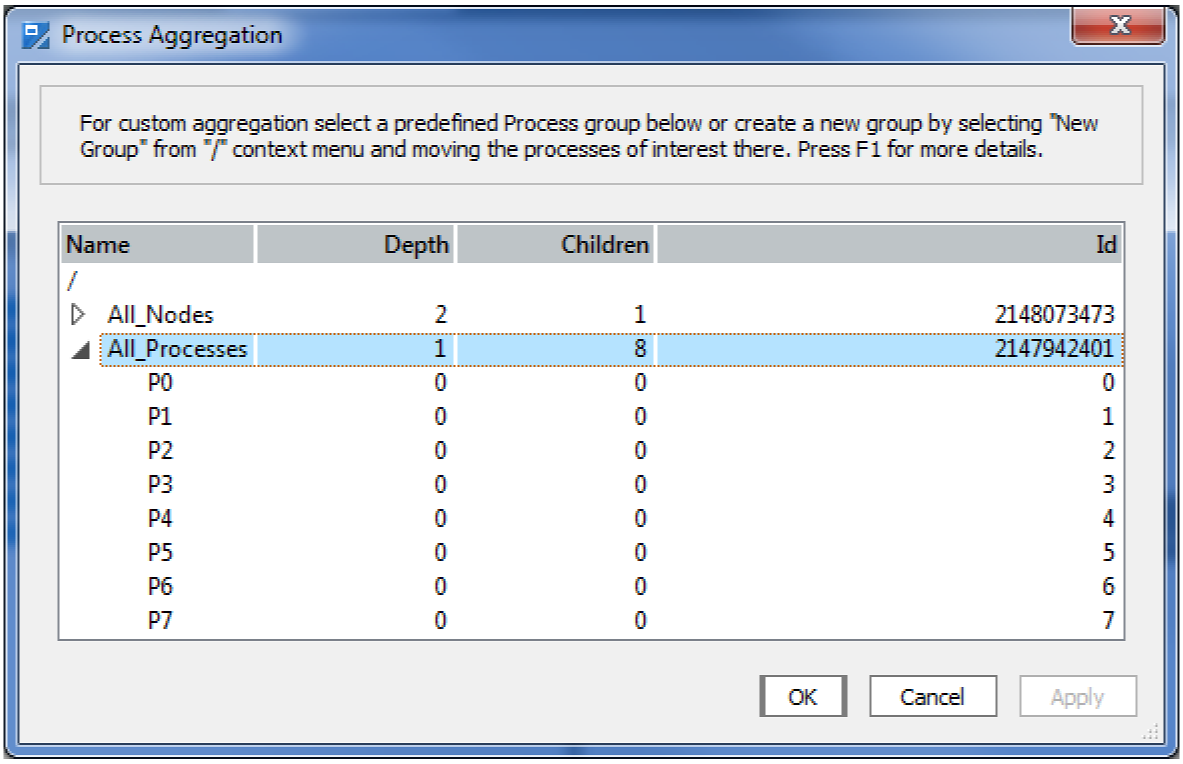
Process Aggregation

Use the Process Aggregation Dialog Box to:

- Select a process group (or function group in general) for process aggregation
- Create new groups beyond the ones that are provided by default

Open the Process Aggregation dialog box through **Advanced > Process Aggregation** or using the toolbar button.

Group definitions are stored in the `.itarc` file in your home directory (for an English Microsoft* Windows XP* installation, this is similar to *`C:\Documents and Settings\%username%\.itarc`*). See Configuration Dialogs for other ways to save, edit and load configuration information.



For custom aggregation select a predefined Process group below or create a new group by selecting "New Group" from "/" context menu and moving the processes of interest there. Press F1 for more details.

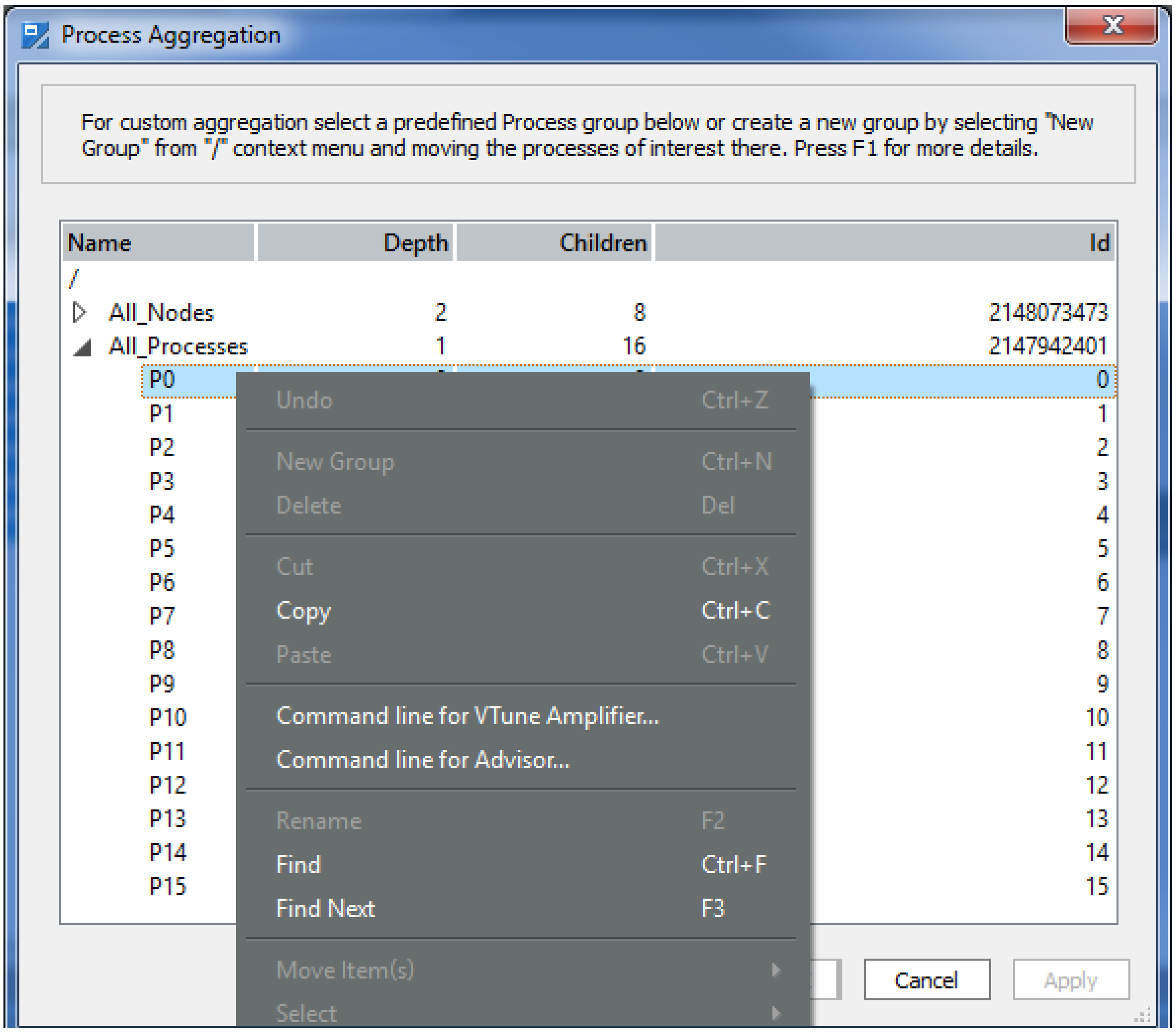
Name	Depth	Children	Id
/			
▷ All_Nodes	2	1	2148073473
◢ All_Processes	1	8	2147942401
P0	0	0	0
P1	0	0	1
P2	0	0	2
P3	0	0	3
P4	0	0	4
P5	0	0	5
P6	0	0	6
P7	0	0	7

To select a process group for aggregation, select the group by using the mouse or the cursor/arrow keys and press **Apply** or **OK**.

NOTE

The dialog box only accepts a single, non-empty process group that contains each of its functions not more than once.

Right-click an item in the editor to bring up a context menu with several entries:



For custom aggregation select a predefined Process group below or create a new group by selecting "New Group" from "/" context menu and moving the processes of interest there. Press F1 for more details.

Name	Depth	Children	Id
/			
All_Nodes	2	8	2148073473
All_Processes	1	16	2147942401
P0			0
P1			1
P2			2
P3			3
P4			4
P5			5
P6			6
P7			7
P8			8
P9			9
P10			10
P11			11
P12			12
P13			13
P14			14
P15			15

Option:	**Keyboard Shortcut:**	**Description:**
Undo	`Ctrl+Z`	Revert the last actions (delete, cut, move, copy and others)
New Group	None	Creates a new group as a child of the clicked item
Delete	None	Remove an item. This entry is disabled for items that originate of the trace file. Only user-created items are deleted.
Rename	F2	Rename user-created items
Find	`Ctrl+F`	Open the **Find** dialog box
Find Next	F3	Search for the next match if search was started before
Move Item(s)	None	Move several selected siblings up or down relative to their unselected siblings through a submenu. It is much more convenient to use the keyboard shortcuts shown in the submenu instead of the menu itself.

Select	None	Open a submenu to select subsets of the tree under the clicked item. The entries provided distinguish between selecting processes, selecting process groups or both, and if the selection should include only the direct children or all descendants.
Command line for VTune Amplifier/Advisor...	None	Open the **Command line for Intel® VTune™ Amplifier XE/Intel®Advisor XE** dialog box

Another way of editing is to drag a group, a process, or the entire current selection and to drop it into a target group. This can result in groups that contain the same function twice or more. In the group hierarchy, such groups are accepted for storage but not for aggregation. Empty groups are deleted automatically when the dialog box is closed.

See Also

Aggregation
Configuration Dialogs
Find Dialog
Command line for Intel® VTune™ Amplifier XE and Intel® Advisor XE Dialog Box

Comparison Mode

In comparison mode (see Comparison of two Trace Files) the dialog also contains a label and a combo box that enables you to apply the chosen aggregation to the other trace file shown in the View.

Use the **If match found** option to find a process group of the same name in the other file and select it.

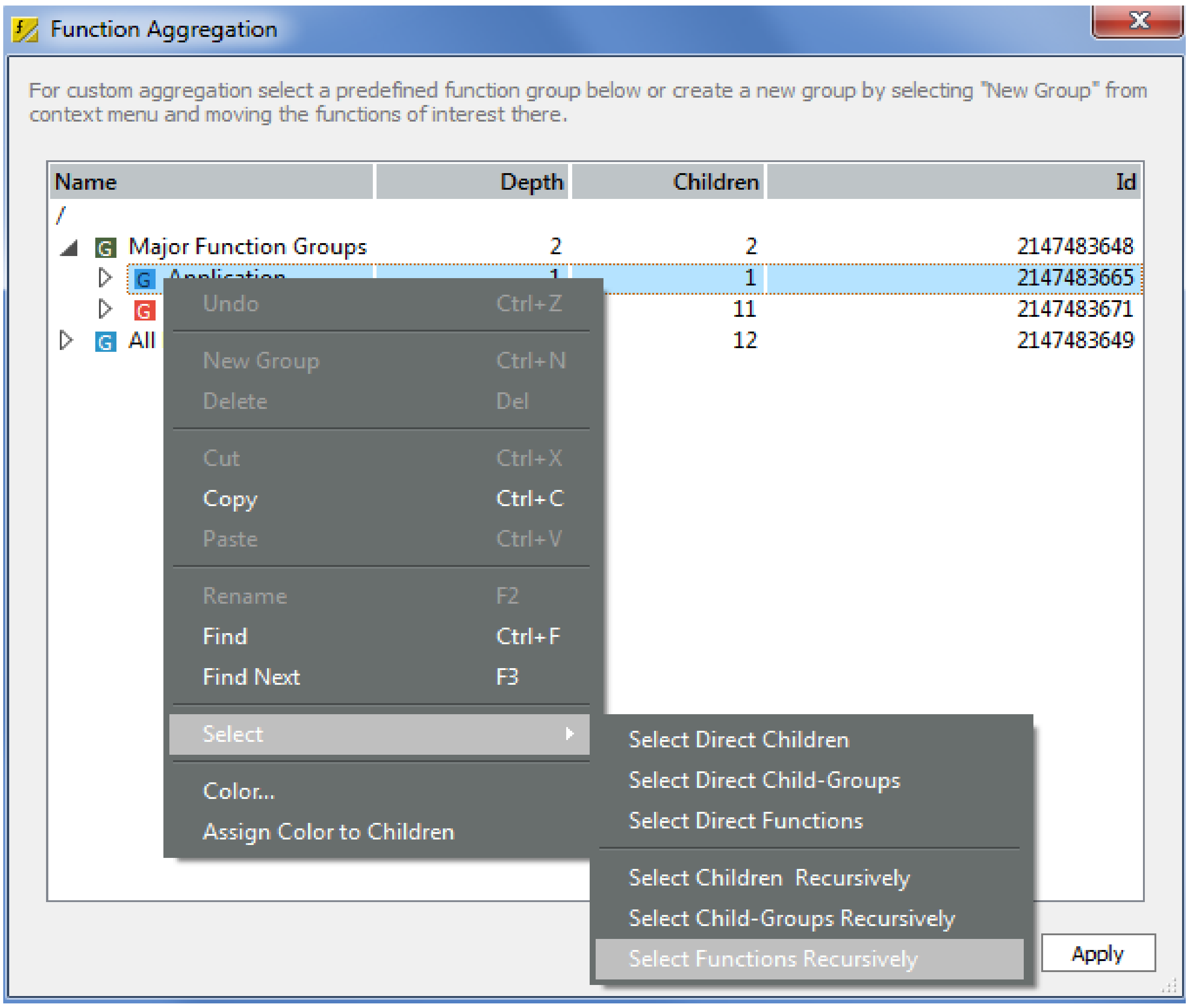
Function Aggregation

In most respects the Function Aggregation works exactly like the Process Aggregation. It enables you to edit function group definitions and to choose a function group for aggregation in a View. The only addition is that you can assign colors to functions and function groups through the context menu\.

Open the Function Aggregation dialog box through **Advanced > Function Aggregation** or using the toolbar button.

To see the full function name, hover your mouse over a group entry and see a tooltip. The full function name (or static path) reflects the original definition of the function or group as stored in the trace file.

You can also manage function groups through the context menu of the Function Aggregation dialog box:



For example, to change the color of any of the functions, select the **Color** option from the context menu and apply a new color to the given function. To assign the color of the parent node to all the children, use the **Assign Color to Children** entry of the context menu. This entry is enabled only if a parent node is selected.

See Also

Process Aggregation
Function Group Color Editor

Comparison Mode

In comparison mode (see Comparison of two Trace Files) the dialog also contains a combo box that enables you to apply the chosen aggregation to the other trace file shown in the View.

Use the **If match found** option to find a matching function group according to the matching rules explained in Mapping of Functions and choose it. It works well for predefined groups. If no match is found, the aggregation for the other file remains unchanged.

Use the **Always (Create matching FGroup)** option to find a match. If a match is not found, Intel® Trace Analyzer creates a matching function group in the name space of the other file.

NOTE

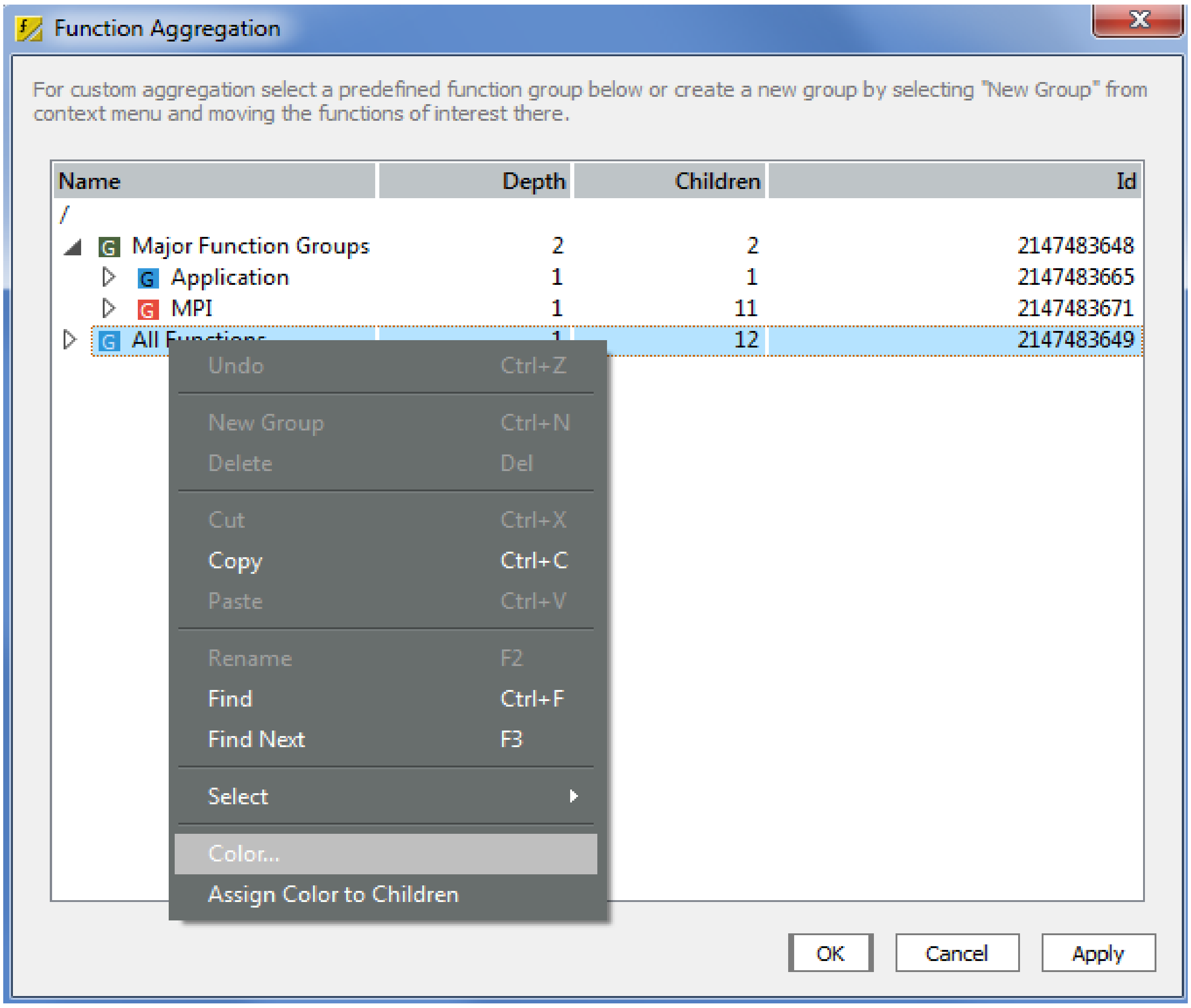
This operation creates a function group that mimics the hierarchical structure of the original group and contains only the functions that are present in both trace files.

Function Group Color Editor

The Function Group Color Editor is the Function Aggregation in a restricted mode that only enables you to edit the colors of functions and function groups.

You can access the Function Group Color Editor through the Event Timeline settings dialog box or through the context menu of the Function Aggregation dialog box context menu.

The context menu of the Function Group Color Editor has a **Find** entry and a **Find Next** entry, both of which are explained in Process Aggregation and also a **Color** entry and an **Assign Color to Children** entry, which are explained in Function Aggregation.


Function Aggregation

For custom aggregation select a predefined function group below or create a new group by selecting "New Group" from context menu and moving the functions of interest there.

Name	Depth	Children	Id
/			
Major Function Groups	2	2	2147483648
Application	1	1	2147483665
MPI	1	11	2147483671
All Functions	1	12	2147483649

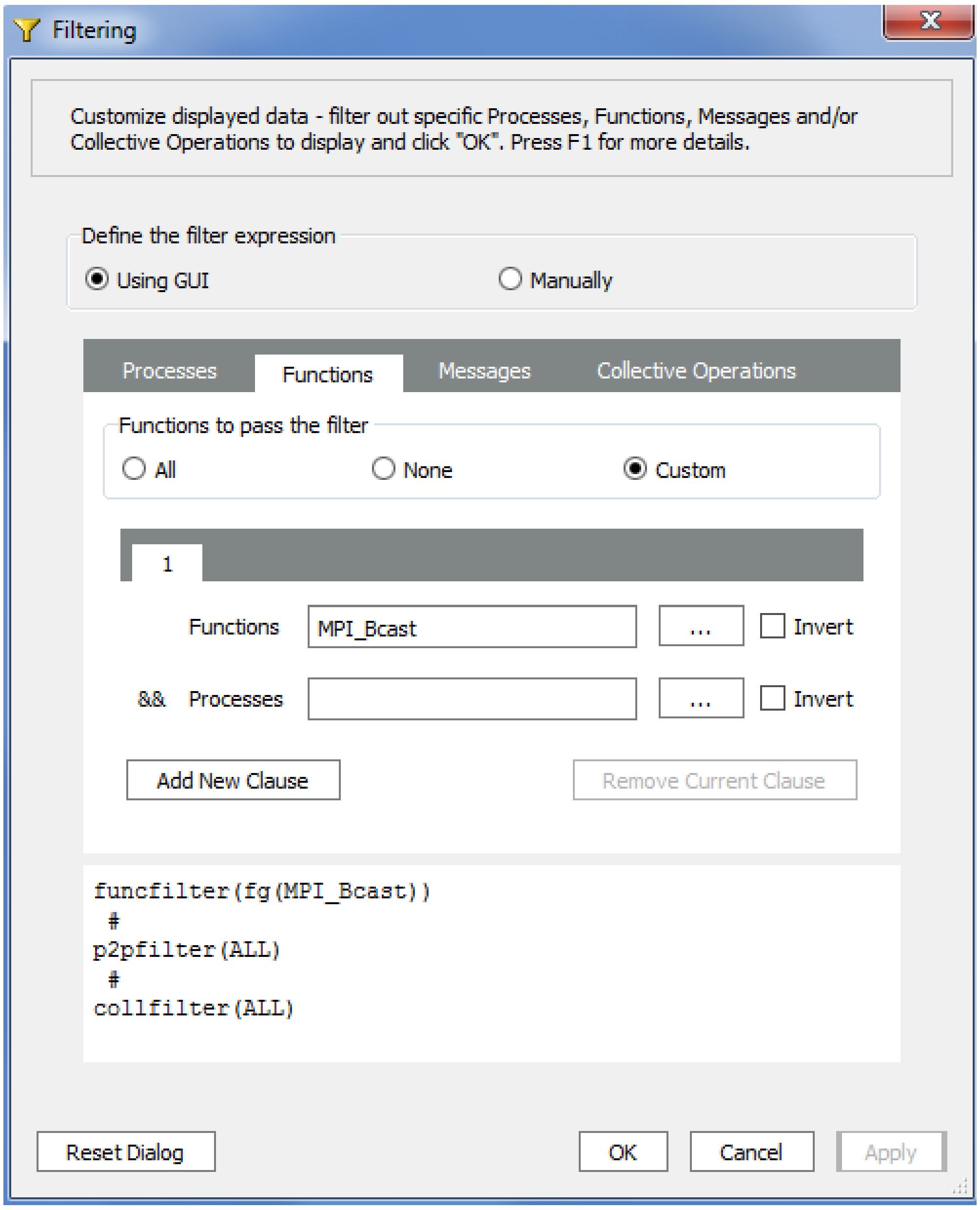
See Also

Process Aggregation
Function Aggregation
Event Timeline
Event Timeline Settings

Filtering Dialog Box

Use the filtering dialog box to specify filter expressions that describe which function events, messages and collective operations are to be filtered out and which are to be shown.

To access the Filtering dialog box, go to **Advanced > Filtering** or use the toolbar button.


Filtering
Customize displayed data - filter out specific Processes, Functions, Messages and/or Collective Operations to display and click "OK". Press F1 for more details.
Define the filter expression
Using GUI
Manually
Processes
Functions
Messages
Collective Operations
Functions to pass the filter
All
None
Custom
1
Functions
MPI_Bcast
...
Invert
&&
Processes
...
Invert
Add New Clause
Remove Current Clause
Reset Dialog
OK
Cancel
Apply

```
funcfilter(fg(MPI_Bcast))
 #
p2pfilter(ALL)
 #
collfilter(ALL)
```

In the **Definition of Filter Expression** section of this dialog box, select the mode to define the filter expression:

Choose This:	To Do This:
Using GUI Interface (the default)	Generate filter expressions through a graphical interface
Manually	Type in the filter expression directly

You can build a filter expression using either the GUI interface or the manual mode, either way the resulting expression is parsed upon each change.

If the current expression cannot be converted into a proper filter definition, a red warning indicating the reason is displayed.

If the expression uses filter attributes that require bypassing them in memory trace cache, a warning message is presented that indicates the need to process data directly from the trace file and the additional time required for the analysis.

See Also

Tagging and Filtering

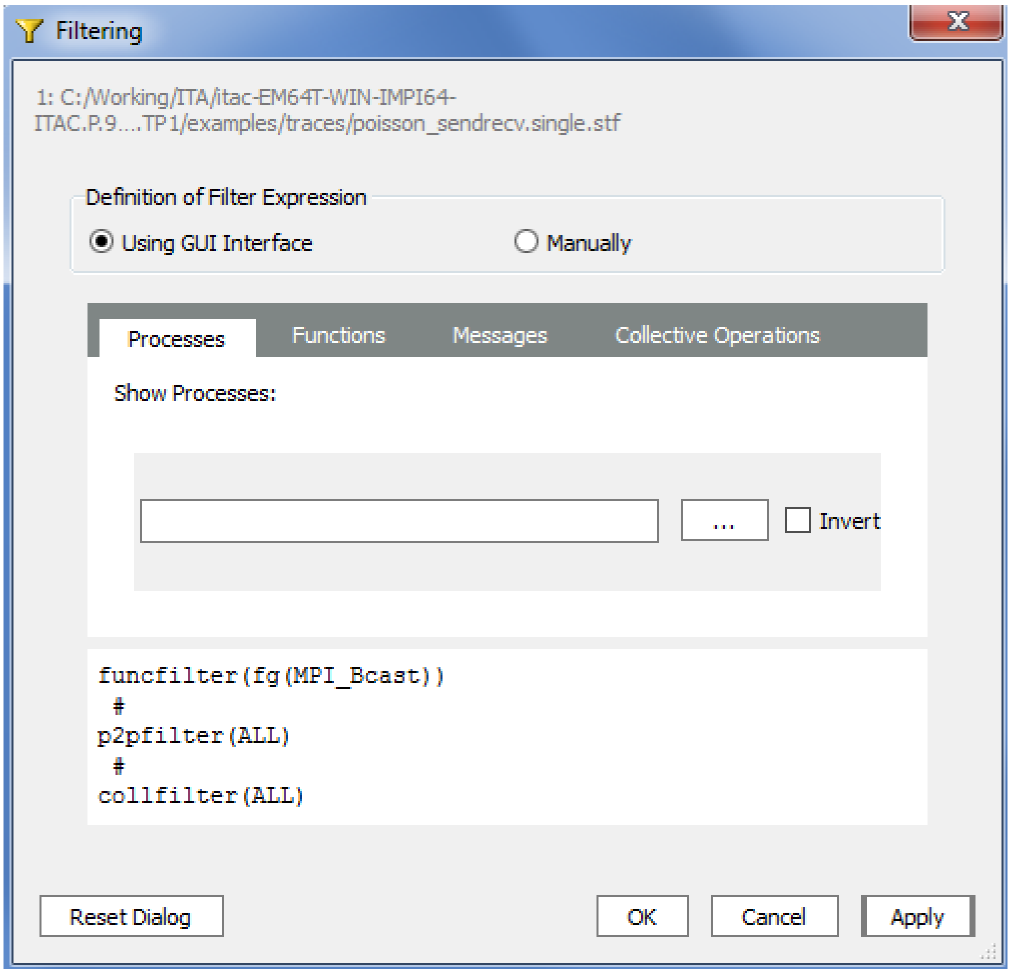
Building Filter Expressions Using Graphical Interface

Use the **Filtering** dialog box to specify filter expressions separately for function events, messages and collective operations under the respective tabs. You can also use the **Processes** tab to restrict the events to a subset of processes.

Some of the text fields in the filter dialog box can take triplets that are of the form `start:stop:incr` or the short form `start:stop` if the increment is one. Such a triplet describes all numbers greater than or equal to start and smaller than or equal to stop and that are of the form `start+incr*n`. The stop value is optional, too. When stop is omitted, all numbers, beginning from start, match.

Processes Tab

Use this tab to specify the processes to pass the filter. This tab has effects on all three sub-expressions: Functions, Messages and Collective Operations.



```
funcfilter(fg(MPI_Bcast))
 #
p2pfilter(ALL)
 #
collfilter(ALL)
```

Use This:	To Do This:
Show Processes	List the desired processes in the text field to see them in the resulting view and filter out all the other processes. The text field can contain a comma-separated list of process ids, process group ids, triplets thereof, unquoted or double-quoted names of threads, processes or process groups. Names given match all equally named processes/groups.

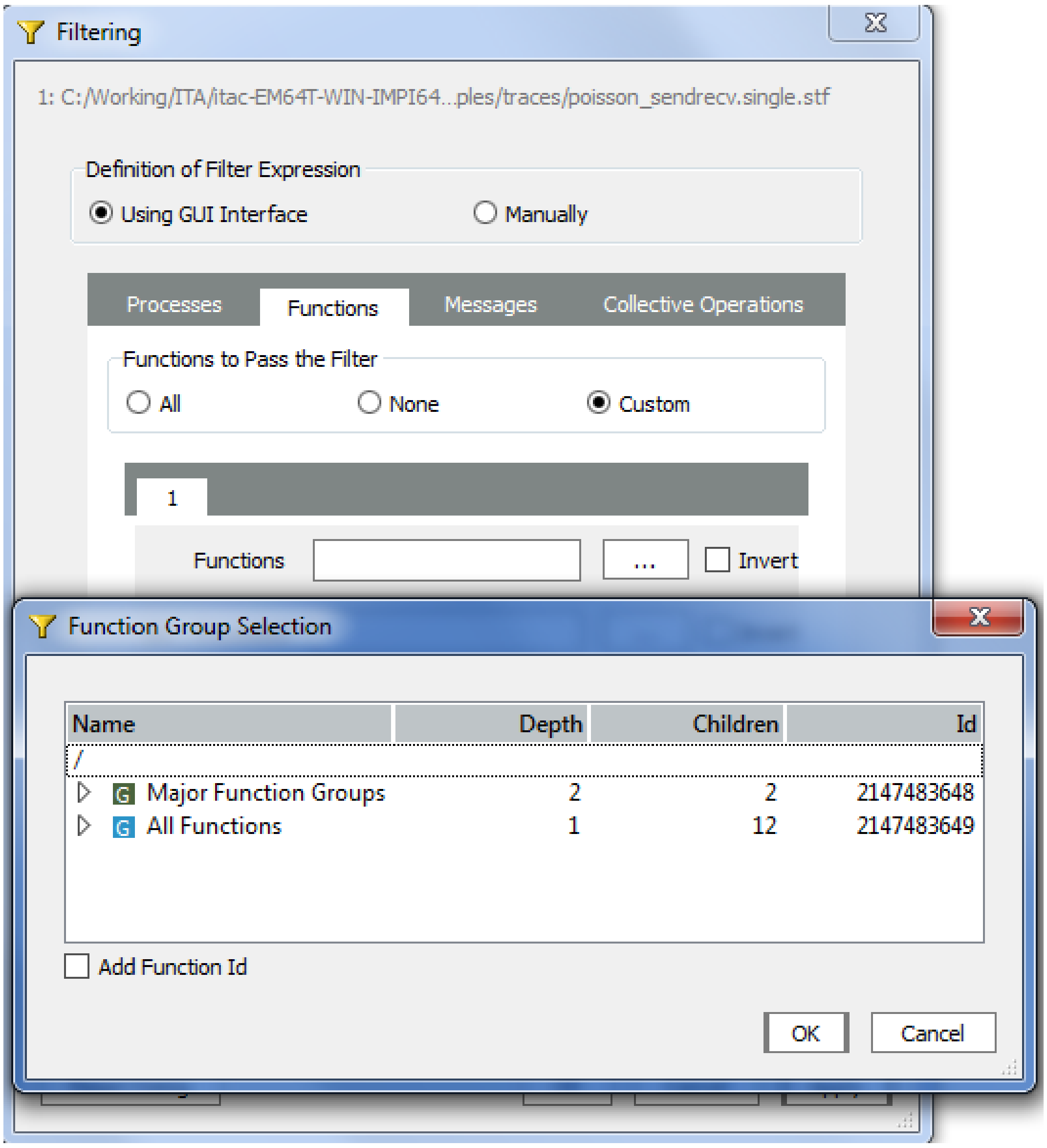
Three-dot-labeled button	Open the **Process Group Selection** dialog box and select one or more processes or process groups.
Invert	Swap the selection. For example, if you want all processes except one to pass the filter, select the process to be filtered out and use the **Invert** check box. It is a logical NOT and it is marked with an exclamation mark before the predicate in the filter expression.

See the resulting filter expression below the filter clauses. To reuse the expression elsewhere, select and copy the expression in the manual mode of the filtering dialog box.

The context menu of the filter expression contains a Copy (`Ctrl+C`) entry and a Select All (`Ctrl+A`) entry.

Functions Tab

In the **Functions** tab, use the **All**, **None** and **Custom** radio buttons to select the mode of filtering.



Definition of Filter Expression: [✓ Using GUI Interface] Manually

Functions to Pass the Filter: All None [✓ Custom]

Name	Depth	Children	Id
/			
Major Function Groups	2	2	2147483648
All Functions	1	12	2147483649

To enable the filter clause tab, select the **Custom** radio button. The filter clause consists of the following entries:

Entry	**Description**

Functions	Use this tab to add functions to the filter. The text field can contain a comma-separated list of functions or function group ids, triplets thereof, unquoted or double-quoted names of functions or function groups that describe functions passing the filter. All given names match all equally named functions/groups. Press the three-dot-labeled button to open the **Function Group Selection** sub-dialog and choose the function names from the list. Check the **Add Function Id** check box to write the name and Id of a selected function into the text field. This is useful to resolve ambiguities of function/group names: if only the name is written into the text field, all functions with this name pass the filter; if the name is replaced by an id, only the function with this id passes the filter, not other functions with the same name.
Processes	Use this tab to add processes to the filter. The text field can contain a comma-separated list of process ids, process group ids, triplets thereof, unquoted or double-quoted thread, process or process group names that describe processes and threads in the functions passing the filter. All given names match all equally named processes/groups. Press the three-dot-labeled button to choose processes from a list. In the **Process Group Selection** sub-dialog, check the **Add Process Id** check box to write the process name and Id into the text field.

To add another filter clause, press the **Add New Clause** button. To remove an existing filter clause tab, use the **Remove Current Clause** button. Clauses are connected by a logical OR, while attributes from the same tab are connected by a logical AND.

Messages

To enable the filter clause tab in the Messages tab, select the **Custom** radio button. The filter clause tab has the following entries:

Entry	**Description**
Communicator	The text field can contain a comma-separated list of communicator ids, unquoted communicator names or communicator names in double quotes that pass the filter. Press the three-dot-labeled button to choose from the list.
Tag	The text field can contain a comma-separated list of non-negative integers and triplets that describe tag values that pass the filter.
Processes	Makes a message pass the filter if either the sender or the receiver matches. Analogous to the logical OR of **Sender** and **Receiver**.
Sender	The text field can contain a comma-separated list of process ids, process group ids, triplets thereof, unquoted or double-quoted names of threads, processes or process groups that describe processes and threads in the message sender that make the message pass the filter. Given names match all equally named processes/groups. Press the three-dot-labeled button to choose from the list.

Receiver	Analogous to **Sender**.
Ranks	Makes a message pass if either the sender or the receiver matches. Analogous to the logical OR of **Rank of Sender** and **Rank of Receiver**.
Ranks of Sender	The text field can contain a comma-separated list of nonnegative integers and triplets that describe sender ranks (in the MPI communicator) that can pass the filter.
Ranks of Receiver	Analogous to **Rank of Sender**.
Message Size	The text field can contain a comma-separated list of non-negative integers and triplets that describe message sizes (in bytes) that pass the filter.
Start Time	The text field can contain a comma-separated list of non-negative integers and triplets that describe start time (in ticks) of the message that makes the operation pass the filter. Press the three-dot-labeled button to enter/edit the time in ticks or seconds (default depending on the View current time unit).
End Time	Analogous to **Start Time**.
Duration	The text field can contain a comma-separated list of non-negative integers and triplets that describe the duration (in ticks) of the message that makes the operation pass the filter. Press the three-dot-labeled button to enter/edit the duration (shown as a time interval) in ticks or seconds (default depending on the View current time unit).

Collective Operations

To enable the filter clause tab in the **Collective Operations** tab, select the **Custom** radio button. The filter clause tab has the following entries:

Entry	Description
Communicator	The text field can contain a comma-separated list of communicator ids, unquoted communicator names or communicator names in double quotes that pass the filter. Press the three-dot-labeled button to choose from the list.
Collective Operation	The text field can contain a comma-separated list of unquoted or double-quoted names of collective operations that pass the filter. Press the three-dot-labeled button to choose from the list.
Processes	The text field can contain a comma-separated list of process ids, process group ids, triplets thereof, unquoted or double-quoted names of threads, processes or process groups that pass the filter. Given names match all equally named processes/groups. Press the three-dot-labeled button to choose from the list.

Root	The text field can contain a comma-separated list of process ids, process group ids, triplets thereof, unquoted or double-quoted names of threads, processes or process groups that describe processes and threads serving as Root in the operation that passes the filter. Given names match all equally named processes/groups. Press the three-dot-labeled button to choose from a list.
Rank of Root	The text field can contain a comma-separated list of non-negative integers and triplets that describe root ranks of the operation that passes the filter.
Transferred Volume	The text field can contain a comma-separated list of nonnegative integers and triplets that describe all volumes (in total bytes per operation) that make a collective operation pass the filter.
Start Time	The text field can contain a comma-separated list of non-negative integers and triplets that describe start time (in ticks) of the operation that passes the filter. Press the three-dot-labeled button to enter/edit the time in ticks or seconds (default depending on the View current time unit).
End Time	Analogous to **Start Time**.
Duration	The text field can contain a comma-separated list of non-negative integers and triplets that describe the duration (in ticks) of the operation that passes the filter. Press the three-dot-labeled button to enter/edit the duration (shown as a time interval) in ticks or seconds (default depending on the View current time unit).

Building Filter Expressions Manually

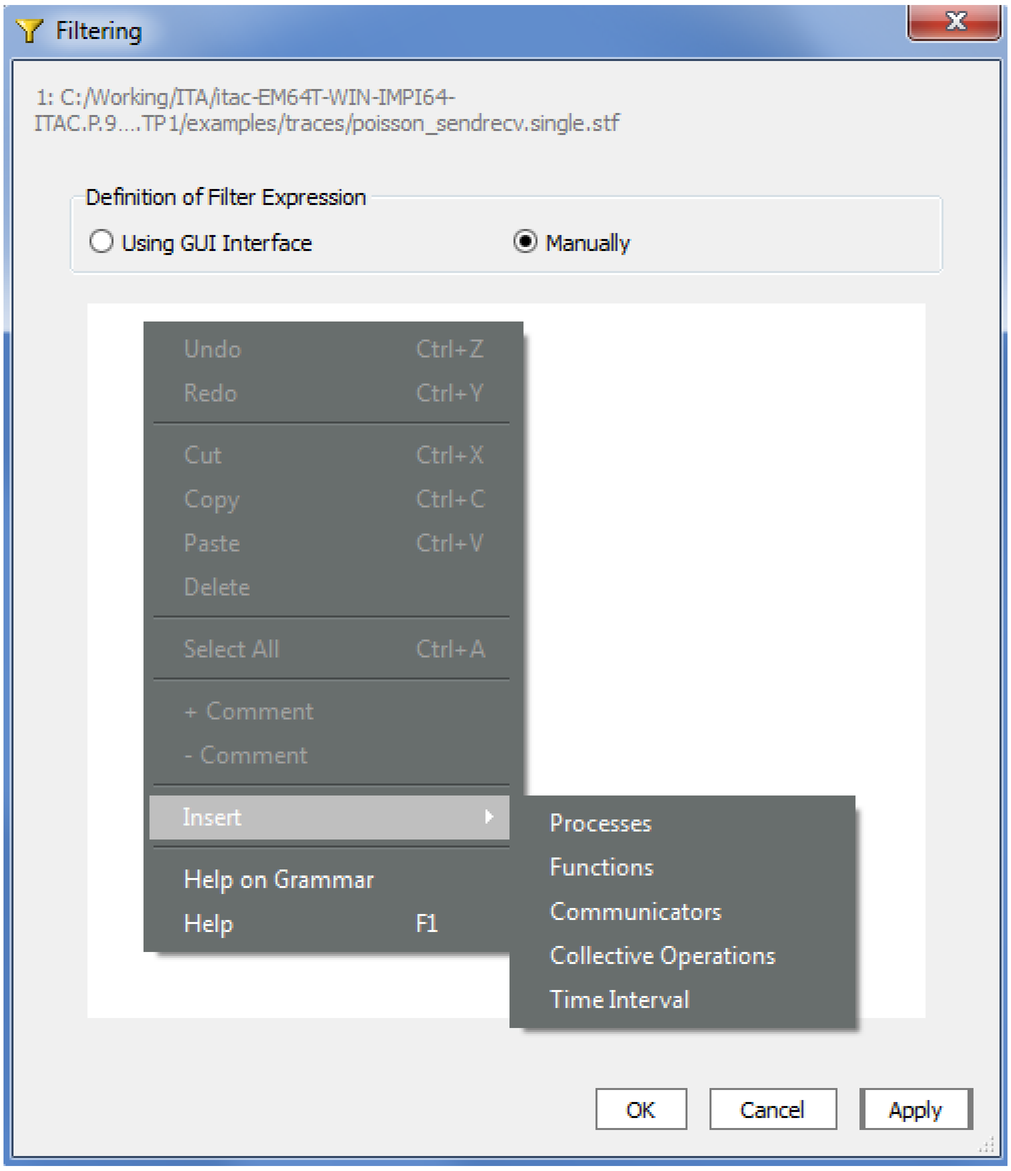
In the Manual mode, construct any filter expression that is valid as described by the expression grammar in the Filter Expression Grammar section.

Use the context menu entries to select processes, functions, communicators and collective operations from a dialog box in the same way as from the graphical interface and to insert them into the expression at the current cursor position.

The percentage sign **%** inserts single line comments and there are context menu entries to comment out (or in) selected text blocks.

There is no operator precedence in the Intel® Trace Analyzer; the expressions are evaluated from left to right. However, you can use parentheses if needed.

Context Menu of the Filtering Dialog Box in Manual Mode:



Filter Expressions in Comparison Mode

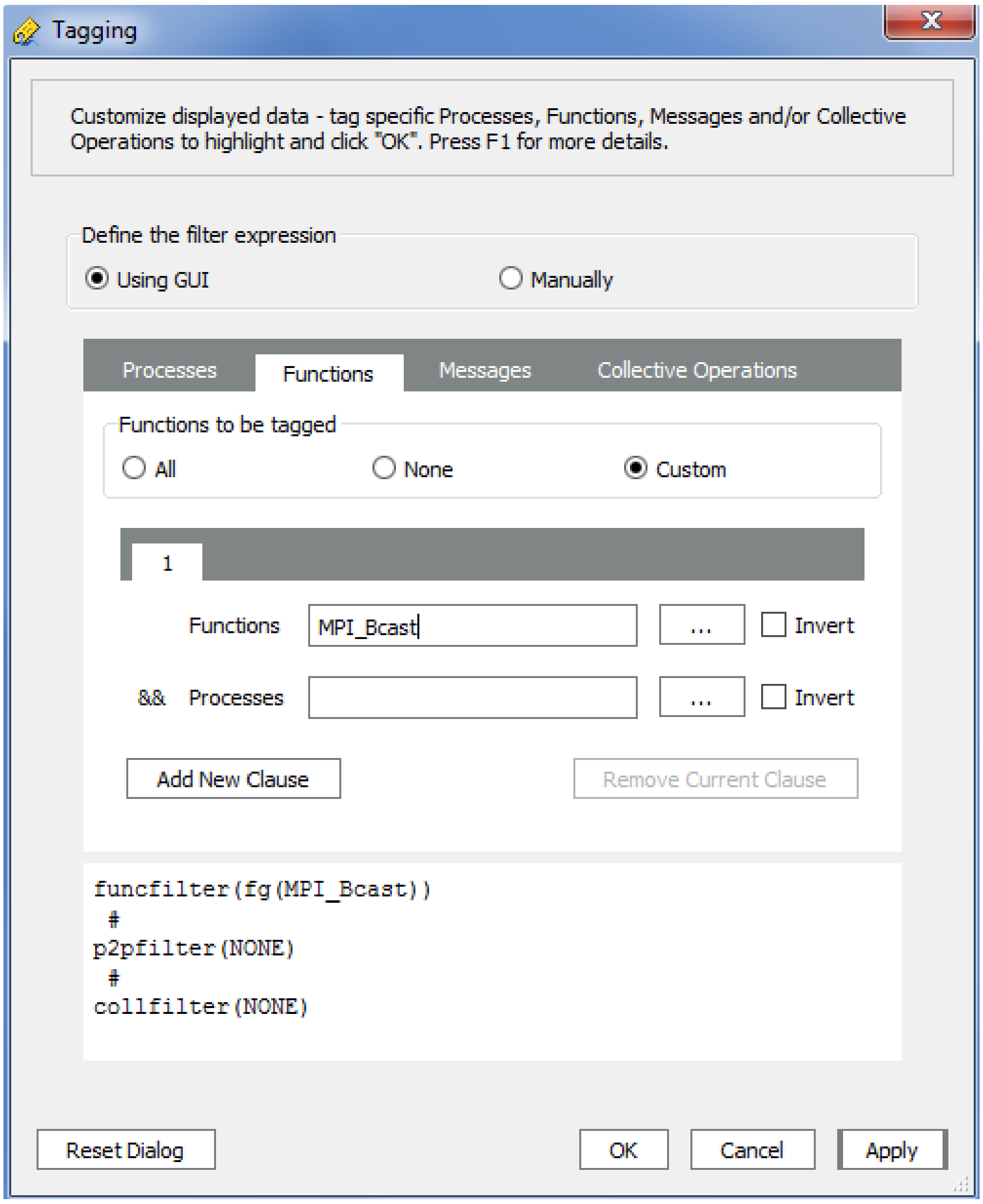
In the Comparison mode (see Comparison of two Trace Files), the Filtering dialog has an additional label and a combo box that enables you to apply the chosen filter expression to the other trace file shown in the View.

In the combo box, choose the **If expression is valid in other file** option to check if the resulting expression is valid in the name space of the other file. If so, the filter applies the expression as if you had manually typed it in the other file filter dialog box. Any previous input in the target dialog box is overwritten.

Tagging Dialog Box

For explanation of the filtering and tagging concept refer to the Tagging and Filtering section.

Configure the **Tagging** dialog box in same way as you configure the **Filtering** dialog box. For description, see Filtering Dialog Box.



Customize displayed data - tag specific Processes, Functions, Messages and/or Collective Operations to highlight and click "OK". Press F1 for more details.

Define the filter expression: [✓ Using GUI] Manually

Functions to be tagged: All None [✓ Custom]

Functions: MPI_Bcast ... Invert

&& Processes: ... Invert

```
funcfilter(fg(MPI_Bcast))
 #
p2pfilter(NONE)
 #
collfilter(NONE)
```

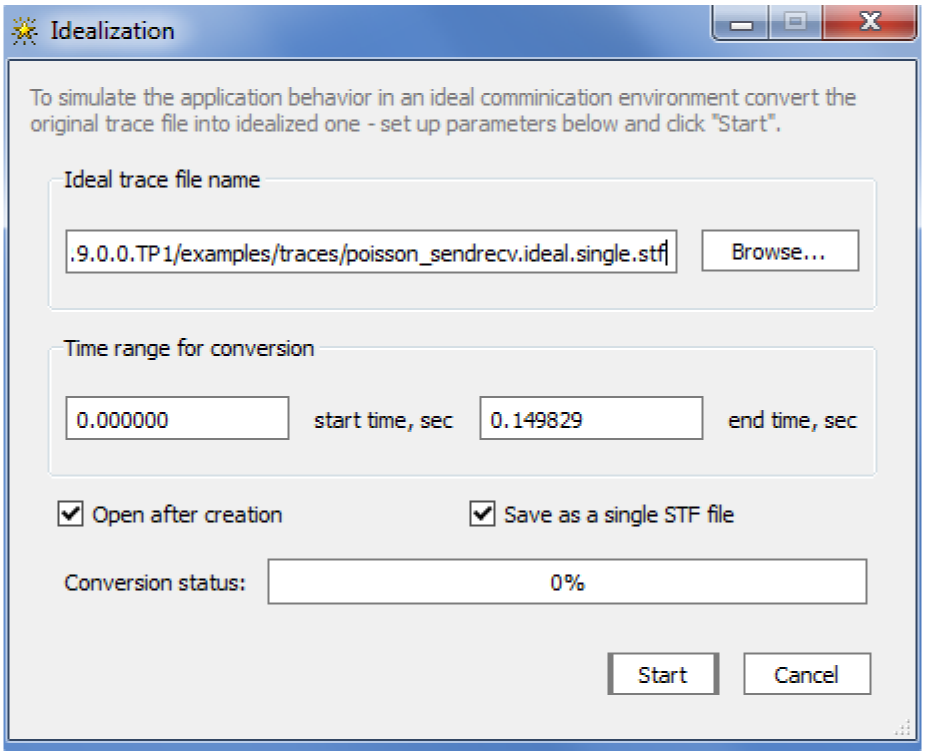
See Also

Filtering Dialog Box
Tagging and Filtering

Idealization Dialog Box

Use the Idealization dialog box to produce an ideal trace.

To access the Idealization dialog box, go to **Advanced > Idealization**, or press the toolbar button.


Idealization

To simulate the application behavior in an ideal comminication environment convert the original trace file into idealized one - set up parameters below and click "Start".

Ideal trace file name: .9.0.0.TP1/examples/traces/poisson_sendrecv.ideal.single.stf

Time range for conversion: 0.000000 start time, sec 0.149829 end time, sec

[✓] Open after creation [✓] Save as a single STF file

Conversion status: 0%

The dialog box contains the following elements:

Entry:	**Description:**
Ideal trace file name	Displays the path to the project input file to be converted and the name of the ideal trace output file. By default, the output file is placed in the same folder as the input file, with the suffix `ideal` added before the `.stf` extension. Type in, or browse to a different location as needed.
Time range for conversion	Specifies the time range of conversion, based on the original trace file time. The time range is specified in seconds, in the **start time** and **end time** fields. The default values are obtained from the original trace.
Open after creation	Check this box to open the newly produced trace in a separate Intel® Trace Analyzer view
Save as a single STF file	Check this box to store the output trace as a single file. By default, the new trace is created in a multiple-file format.
Conversion status	Indicates the percentage of conversion completion. Before clicking the **Start** button, you see the progress label **Ready to Start**. After you click the **Start** button, the label changes to **Converting**.

See Also

Simulating Ideal Communication

Imbalance Diagram Dialog Box

The Imbalance Diagram enables you to compare the performance between a real trace and an ideal trace.

Use the Imbalance Diagram total mode to see the time spent on calculation, interconnect, and imbalance as well as the time distribution between them. The diagram displays the time distribution in different colors.

Use the Imbalance Diagram breakdown mode to see:

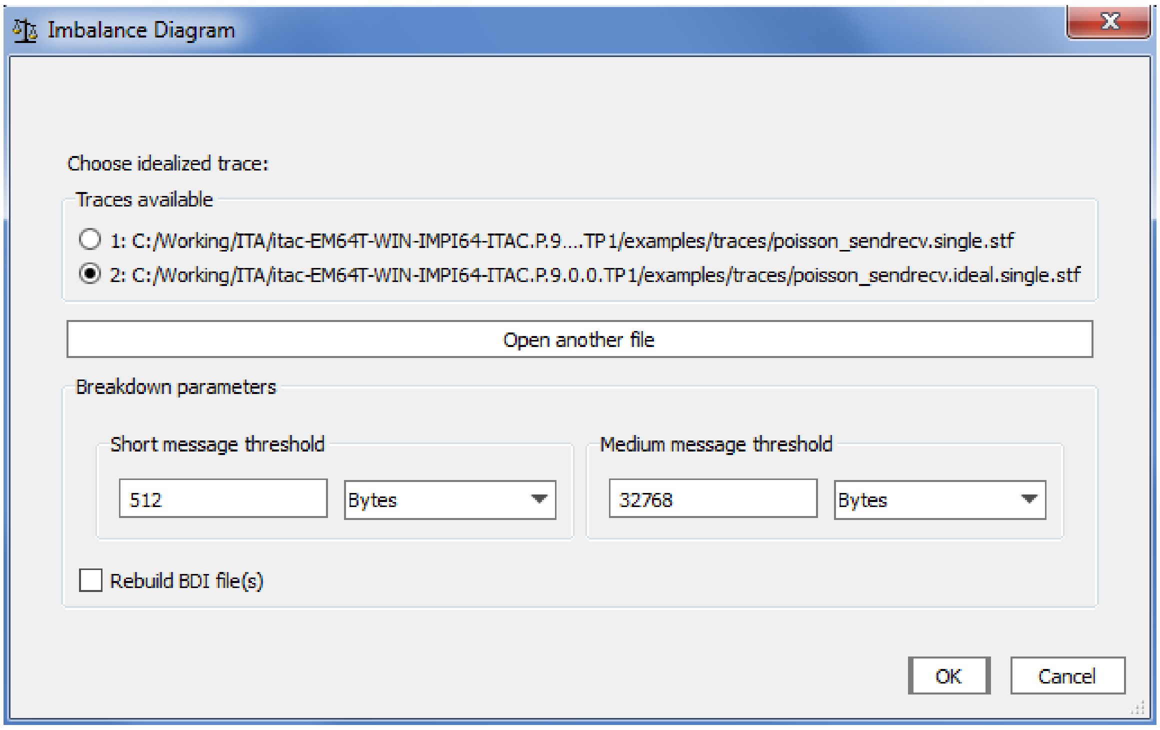
1. how three most time-consuming functions of the application work
2. how much time they spend on three different kinds of messages
3. what the proportion between interconnect and imbalance for each kind of messages is

The messages can be:

1. short, 0-512 bytes
2. medium, 512-32767 bytes
3. large, > 32767 bytes

To compare a real trace with an ideal trace, take the following steps:

1. Go to **Advanced > Imbalance Diagram** or press the toolbar button .
2. In the **Choose Idealized Trace** dialog box, choose a real and an ideal trace that you want to analyze. For example:


Imbalance Diagram

Choose idealized trace:

Traces available

- ○ 1: C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9....TP1/examples/traces/poisson_sendrecv.single.stf
- ● 2: C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9.0.0.TP1/examples/traces/poisson_sendrecv.ideal.single.stf

Open another file

Breakdown parameters

Short message threshold: 512 Bytes

Medium message threshold: 32768 Bytes

☐ Rebuild BDI file(s)

OK Cancel

3. You can specify the following Imbalance Diagram parameters:
   1. Breakdown message thresholds
   2. BDI files rebuilding

   If you have changed the BDI files manually or the files were partially corrupted, select the **Rebuild the BDI files** checkbox.

   The BDI files are the ASCII files and contain all the information needed to create an Imbalance Diagram View. The file name is <*original trace name*>.bdi. The file format is as follows:

```
<version string> <short message threshold> <medium message threshold>
<Function ID> <Collective ID> <Function Name> <Total Function Time>
<Short Messages Time> <Medium Messages Time> <Large Messages Time>
```

4. Click **OK**.

NOTE

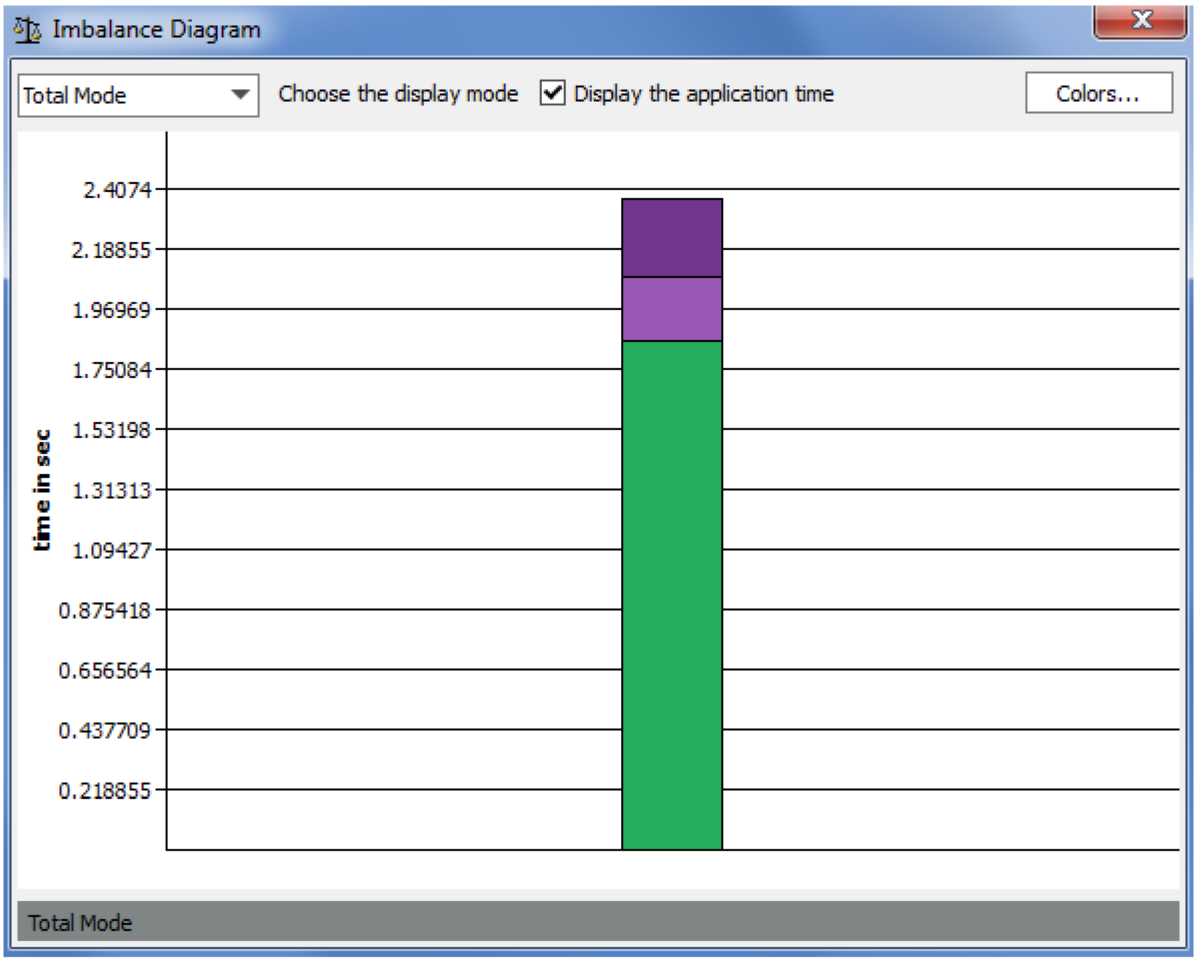
Do not change the BDI file manually. It may result in incorrect feature functioning. You can import the BDI file into a Microsoft Excel* table.

The **BDI File Building** dialog box displays the progress of BDI files building. To stop the BDI building process, click **Cancel**.

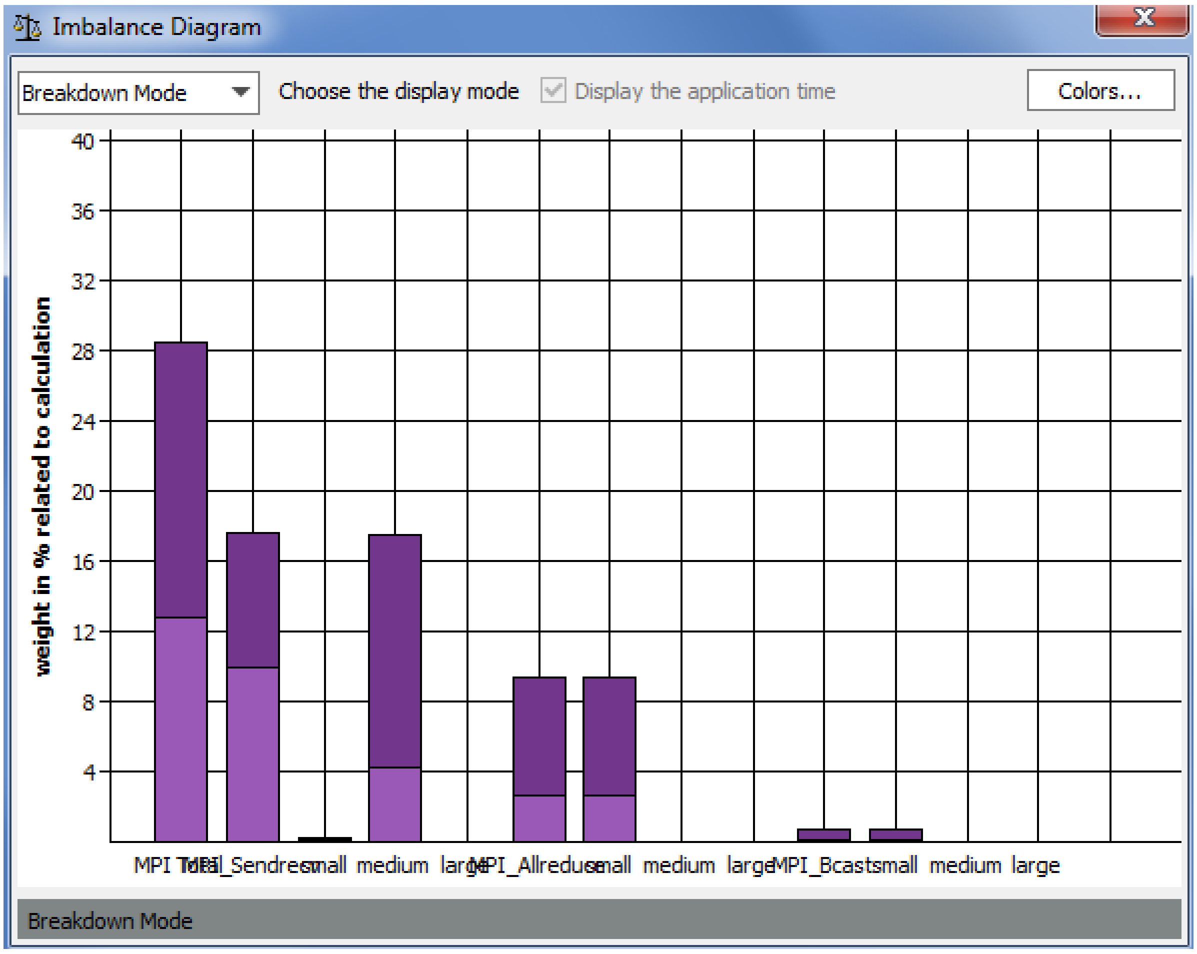
NOTE

If you stop the BDI building, the **Imbalance Diagram** dialog box is not displayed.

In the **Imbalance Diagram** dialog box, specify whether to display and the mode to display (Total mode or Breakdown mode) the application time. For example, Total Mode:

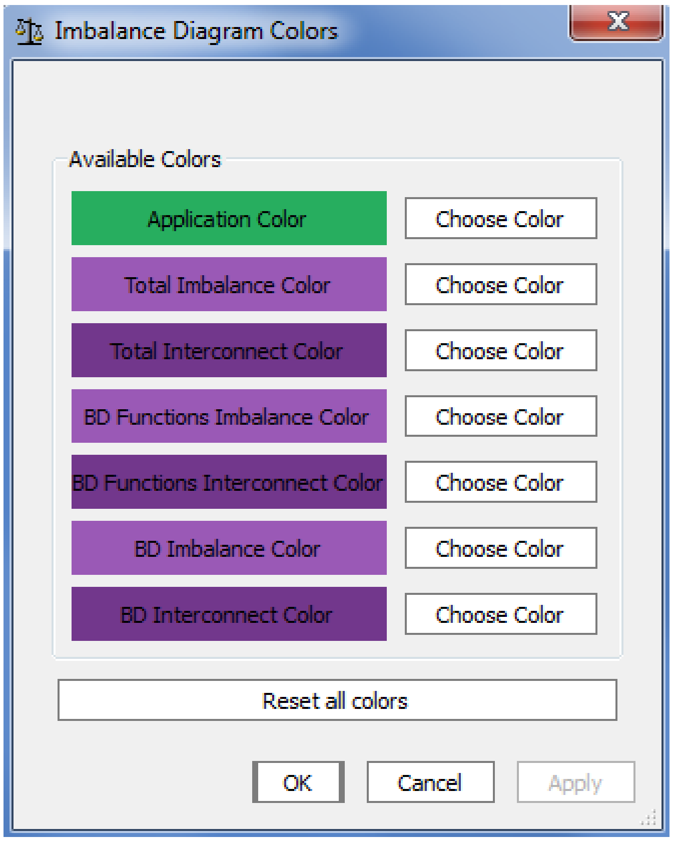


Breakdown Mode:



To change the colors used in the Imbalance Diagram, do the following:

1. In the **Imbalance Diagram** dialog box, click **Colors.**
2. In the **Imbalance Diagram Colors** dialog box, click **Choose Color**. The result will look like this:



3. Choose the colors and click **OK** to save the changes.

NOTE

The new colors are stored in the Intel Trace Analyzer Resource file, so you don't need to change them every time you start the Intel Trace Analyzer. To restore the default colors for all elements, click **Reset all colors**.

See Also

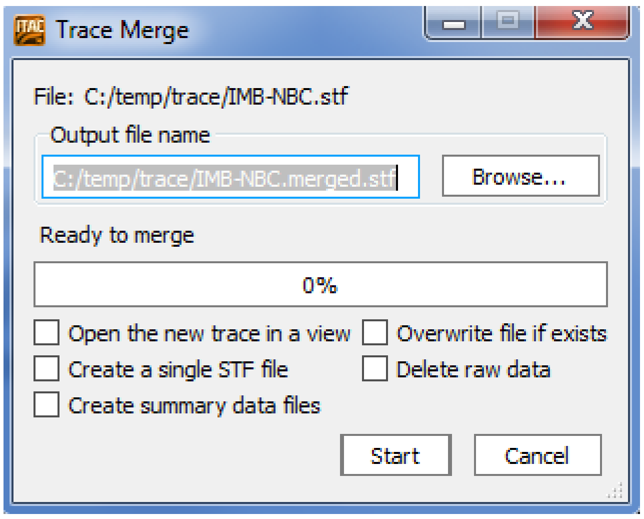
Idealization Dialog Box

Trace Merge Dialog Box

When you are opening a raw trace in Intel® Trace Analyzer, you are prompted to merge the trace. If you click Yes, the Trace Merge dialog box appears.

NOTE

To generate a raw trace, you should have the Intel® Trace Collector setting VT_KEEP_RAW_EVENTS enabled. For details, see the *Intel® Trace Collector User and Reference Guide*.



The dialog box contains the following elements:

Select This:	**To Do This:**
File	See the path of the filename to be merged
Output file name	Contains the name of the output file for a merged trace. By default, the output file is placed in the same folder of the input file, with the suffix `merged` added before the `.stf` extension. Type in, or browse to a different location as needed.
Progress bar	See the percentage of merging done. Before clicking the **Start** button, the progress label is **Ready to Merge**. After clicking the **Start** button, the label changes to **Initializing**, and then to **Merging**.
Open the new trace in a view	Check this box to open the newly produced trace in a separate Intel® Trace Analyzer view
Create a single structured trace format	Check this box to store the resulted trace as a single file. By

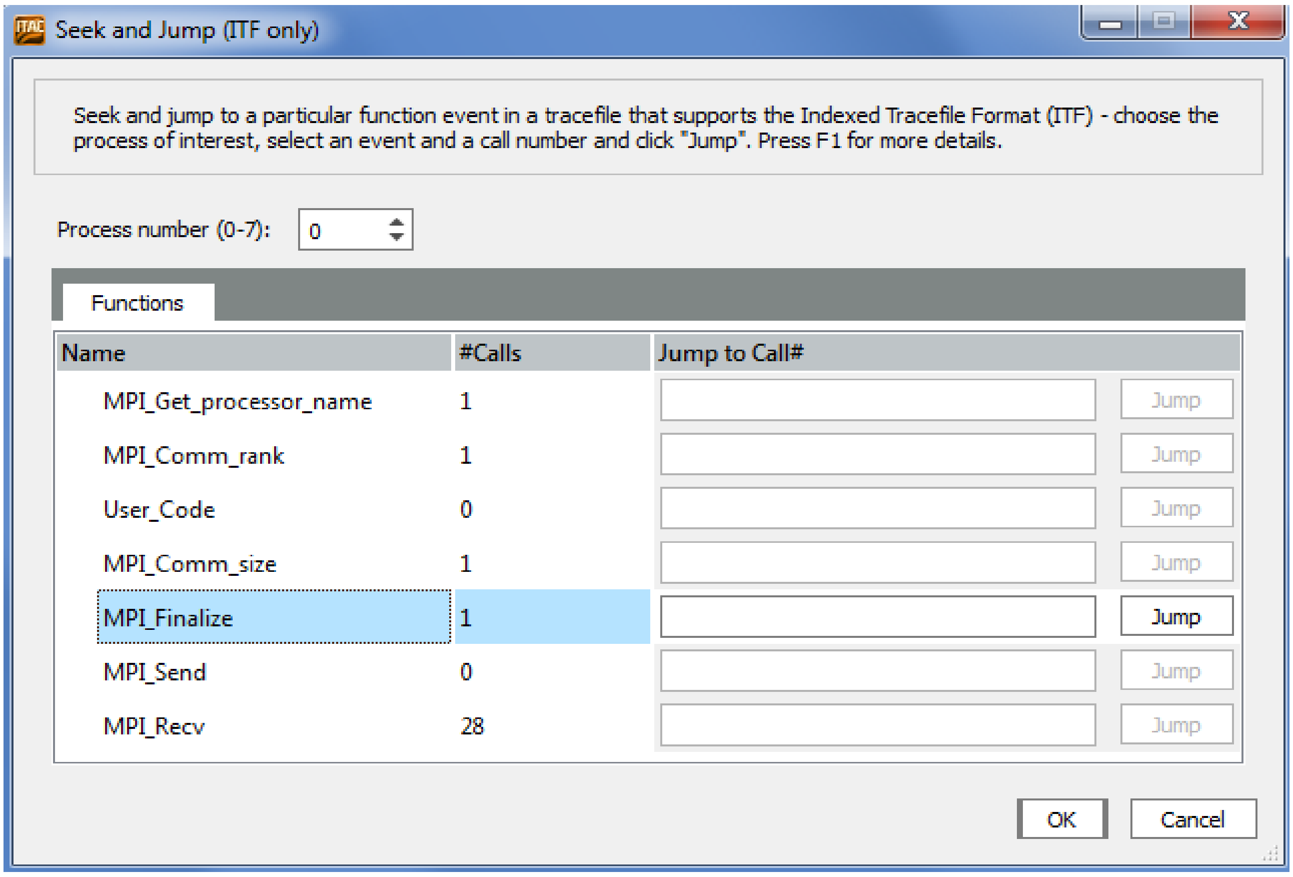
(STF) file	default, the new trace is created in the multiple-file format.
Overwrite file if exists	Check this box if you want to overwrite existing output file
Delete raw data	Check this box if you want to automatically delete the raw trace after merging
Create summary data files	Check this box if you want to create summary data files. The summary data files can help reduce the time when you first open a trace file.

Seek and Jump Dialog Box

The seek and jump function enables you to go directly to a particular function event in the trace.

NOTE

Use this function with the indexed tracefile format (ITF).



Seek and Jump (ITF only)

Seek and jump to a particular function event in a tracefile that supports the Indexed Tracefile Format (ITF) - choose the process of interest, select an event and a call number and click "Jump". Press F1 for more details.

Process number (0-7): 0

Functions

Name	#Calls	Jump to Call#
MPI_Get_processor_name	1	Jump
MPI_Comm_rank	1	Jump
User_Code	0	Jump
MPI_Comm_size	1	Jump
MPI_Finalize	1	Jump
MPI_Send	0	Jump
MPI_Recv	28	Jump

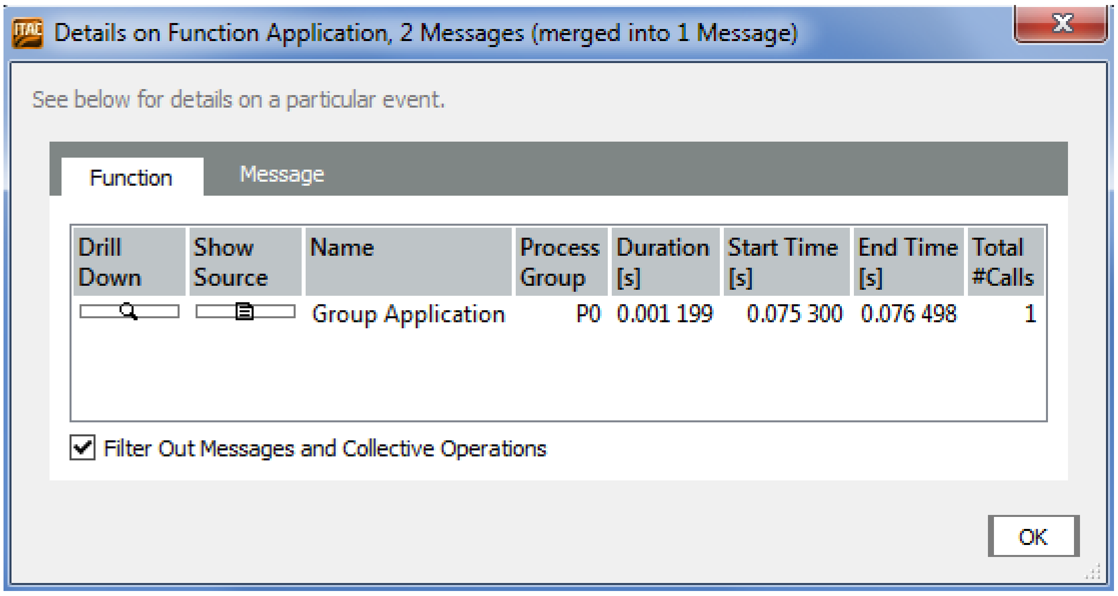
Use the feature to select a particular function event for different processes. Seek and Jump function automatically detects the total number of processes and the number of each function calls in a process, and updates the dialog accordingly to the detected data.

If you select the function, the **Jump** button becomes enabled. Type the desired call number and press **Jump**. The Intel Trace Analyzer zooms all the charts in the **View** to the selected function event.

Details Dialog Box

The dialog box shows detailed attributes of the clicked events. Function events, messages and collective operations are shown in separate tabs. Each tab shows a list of event entries.

This dialog box is available from the context menus of the Event Timeline and the Qualitative Timeline.


Details on Function Application, 2 Messages (merged into 1 Message)

See below for details on a particular event.

Drill Down	Show Source	Name	Process Group	Duration [s]	Start Time [s]	End Time [s]	Total #Calls
		Group Application	P0	0.001 199	0.075 300	0.076 498	1

[✓] Filter Out Messages and Collective Operations

If the column **Count** shows a value greater than one, the event was created by merging several atomic events. Each entry representing a merged event shows a View, which focuses on the data that went into this entry using the drill down button shown next to the entry. This is called a Detail View; it is a full blown View without restrictions. Use the check-box below the list to filter out the other event categories in the Detail View to be opened.

NOTE

Left-click the Details dialog box to reuse an existing Detail View so that the screen is not cluttered so easily. You can also right-click the drill down button to open a new Detail View.

If source code location information is available for an entry, then a button **Show source** appears next to the entry. This button opens a **Source View** dialog box. The source code location is only available up to a certain time interval. If the time interval is set in such a way that there is no enter event for a function, then the **Details** dialog is not aware of the source code location and consequently there will not be a **Show Source** button.

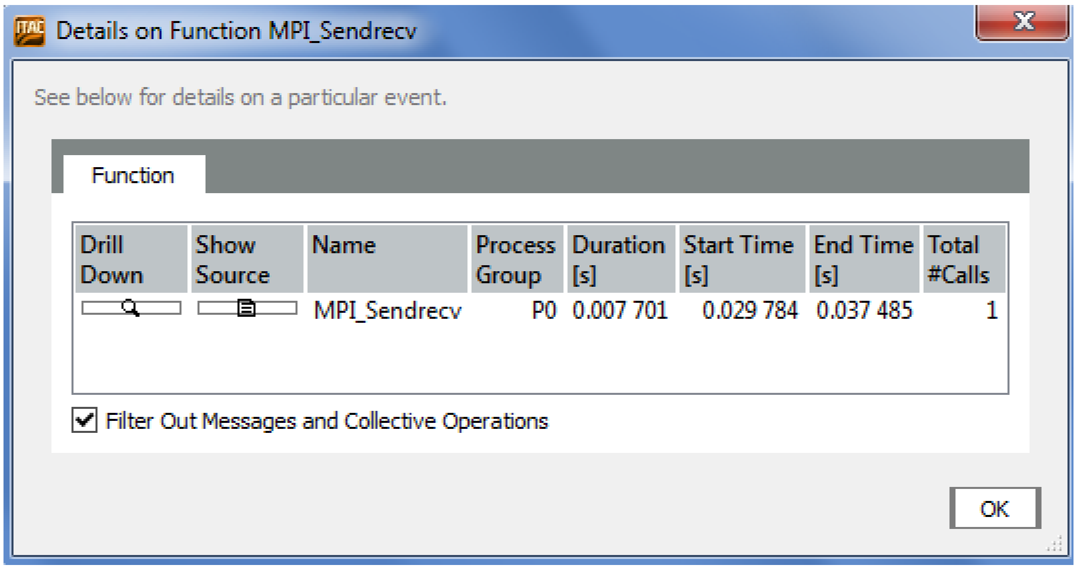
When a **Details** dialog is opened, it preserves all the settings of its View (aggregations, filters, ticks vs. seconds) and the event results for which it is opened. Further changes to the View, like a change in Aggregation, do not affect the dialog contents. When such a change is made to the View, the dialog View label is made bold and an asterisk character ***** appears in front of it. This is to indicate that the details shown are not updated to match the View/Chart. When the Chart or the View from which the Details Dialog originated are closed, the dialog is also closed.

See Also

Level of Detail
Source View Dialog

Detailed Attributes of Function Events

This dialog box provides you with detailed information about function events.


Details on Function MPI_Sendrecv

See below for details on a particular event.

Function

Drill Down	Show Source	Name	Process Group	Duration [s]	Start Time [s]	End Time [s]	Total #Calls
		MPI_Sendrecv	P0	0.007 701	0.029 784	0.037 485	1

[✓] Filter Out Messages and Collective Operations

Use the following entries of this dialog box:

Select This:	**To Do This:**
Name	Specify the name of the selected function. If it represents a Function Group, then it is prefixed with the Group name.
Process Group	Specify the Process, Function or Group in which the selected event occurred
Duration	Show the time spent in a given Function/Group. For coarser resolutions (Count ≥ 1), the value does not reflect the actual time spent in this function but the length of the time interval over which several function events were merged.
Start Time	Show the time when the event entered the Function/Group. For coarser resolutions (Count ≥ 1) the value represents the start of the time interval over which several function events where merged.
End Time	Show the time when the event left the Function/Group. On coarser resolutions (Count ≥ 1) the value represents the end of time the interval over which several function events where merged.
Total #Calls	Give the total number of function calls. It contains all calls covered by the function in the given time interval, including calls to functions other than the function clicked. If the column Total #Calls shows a value greater than one, the event was created by merging several atomic events. Each entry representing a merged event shows a View, which focuses on the data that went into this entry using the drill down button shown next to the entry. This is called a Detail View; it is a full blown View without restrictions. In the dialog box below the list is a check box that allows you to filter out the other event categories in the Detail View to be opened.

NOTE

Sometimes a Function starts before and/or ends after the displayed time interval. In these possible cases, you may see in the Details dialog a less **<** character preceding the numeric value listed under the column Start Time, and a greater **>** character preceding the numeric values listed under the columns **End Time** and **Duration**. These numeric values are the boundaries of the current zoom interval.

Detailed Attributes of Message Events

This dialog box provides you with detailed information about messages.

This dialog box contains the following entries:

Entry:	**Description:**
Sender	The Sender is the Process, Function or Group which sent the message
Receiver	Process, Function or Group which received the message
Duration	Specifies the time taken by the merged operation. It is the difference between Send Time and Receive Time.
Send Time	Specifies the time when the message was sent. If more than 1 message is represented (Count ≥ 1), then the first Send Time of any member in the merge is specified.
Receive Time	This indicates when the message was completely received. If more that 1 message is represented (Count ≥ 1), then the last Receive Time of any member in the merge is specified.
Volume	The total number of bytes sent with selected message(s)
Rate	Indicates the rate at which the bytes are transferred. It is calculated using Volume/Duration.
Count	Specifies the number of messages that are merged into the selection.
Tag	Specifies the MPI tag of the message. If more than one message is merged together, then the tag of the first message is shown.
Communicator Name	The name of the MPI communicator on which the message(s) was (were) sent is specified in this attribute.
Communicator ID	The plain ID of the MPI communicator on which the message(s) was (were) sent is given by this attribute.
Sending Function	The name of the MPI function from which the message(s) was (were) sent is given in this attribute.
Receiving Function	Specifies the name of the MPI function which received the sent message(s).

Detailed Attributes of Collective Operation Events

Each possibly merged collective operation has a header entry which describes the collective operation as a whole. Use the plus button to get a detailed list of the same information per Process/Group. The exact processes or process groups shown depend on the current process aggregation.

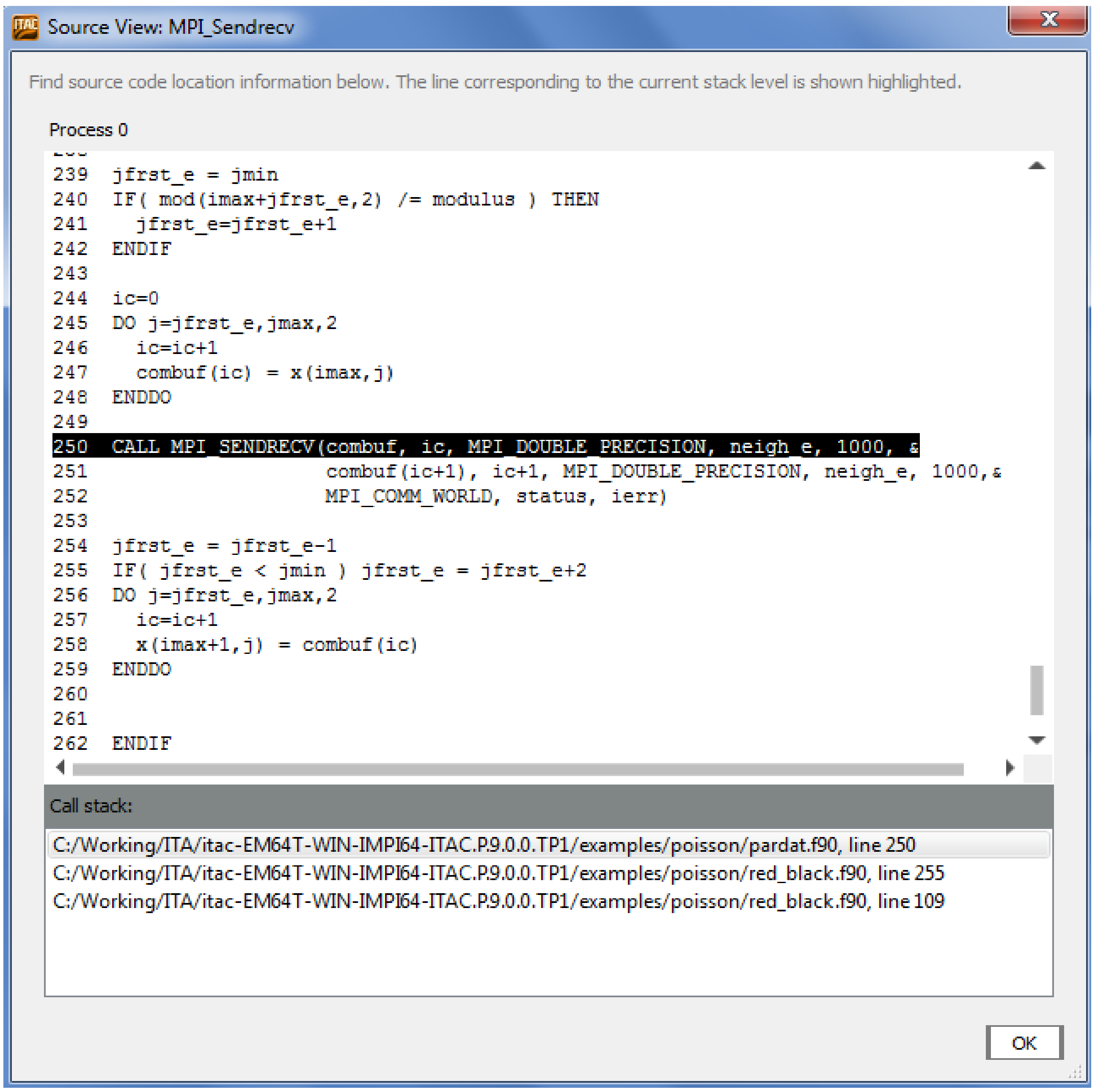
This dialog box contains the following entries:

Entry:	**Description:**
Name/Process	On the per-operation row this column lists the name of the selected operation (**Mixed** if different operation types were merged). On per-process rows it shows the name of the Process/Group.
Duration	Last Time minus First Time
First Time	First time, one of the merged operations was entered.
Last Time	Last time, one of the merged operations was left.
Volume Sent	Number of bytes sent. It is the sum of all bytes sent on all merged operations for the per-process rows. The per-operation row sums up all per-process rows.
Volume Received	Number of bytes received. It is the sum of all bytes received on all merged operations for the per-process rows. The per-operation row sums up all per-process rows.
Count	Number of collective operations that are merged into the selection.
Root	The root process.
Communicator Name	The name of the MPI communicator on which the collective operation(s) was (were) executed.
Communicator ID	The plain ID of the MPI communicator on which the collective operation(s) was (were) executed.

Source View Dialog

Use this dialog box to see the source code.

Open the Source View dialog box from the **Details** dialog box if source code location information is available in the trace file. Use PCTRACE to enable source code location tracing (see the *Intel® Trace Collector User and Reference Guide* for details).



Find source code location information below. The line corresponding to the current stack level is shown highlighted.

Process 0

```
239  jfrst_e = jmin
240  IF( mod(imax+jfrst_e,2) /= modulus ) THEN
241    jfrst_e=jfrst_e+1
242  ENDIF
243
244  ic=0
245  DO j=jfrst_e,jmax,2
246    ic=ic+1
247    combuf(ic) = x(imax,j)
248  ENDDO
249
250  CALL MPI_SENDRECV(combuf, ic, MPI_DOUBLE_PRECISION, neigh_e, 1000, &
251                    combuf(ic+1), ic+1, MPI_DOUBLE_PRECISION, neigh_e, 1000,&
252                    MPI_COMM_WORLD, status, ierr)
253
254  jfrst_e = jfrst_e-1
255  IF( jfrst_e < jmin ) jfrst_e = jfrst_e+2
256  DO j=jfrst_e,jmax,2
257    ic=ic+1
258    x(imax+1,j) = combuf(ic)
259  ENDDO
260
261
262  ENDIF
```

Call stack:

C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9.0.0.TP1/examples/poisson/pardat.f90, line 250
C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9.0.0.TP1/examples/poisson/red_black.f90, line 255
C:/Working/ITA/itac-EM64T-WIN-IMPI64-ITAC.P.9.0.0.TP1/examples/poisson/red_black.f90, line 109

The dialog box consists of a text browser in the center and a list box representing the call stack at the bottom:

Select This:	**To Do This:**
Text browser	View a source file. The line corresponding to the current stack level is shown highlighted in reverse video.
Call stack	Select from the stack levels that were stored with the source code location information. Selecting an entry from this list switches the text browser to the file and line number matching the stack level.

By the default, Intel® Trace Analyzer searches source code files in the current directory and the directory of the current trace file. If no source files are found, you can manually specify the source file to load in the Open File dialog box that appears. The specified path is added to the directory search path. Use the entry **File Default/UserDefines SCLSearchPaths** in the configuration dialog box to specify additional directories to be searched.

See Also

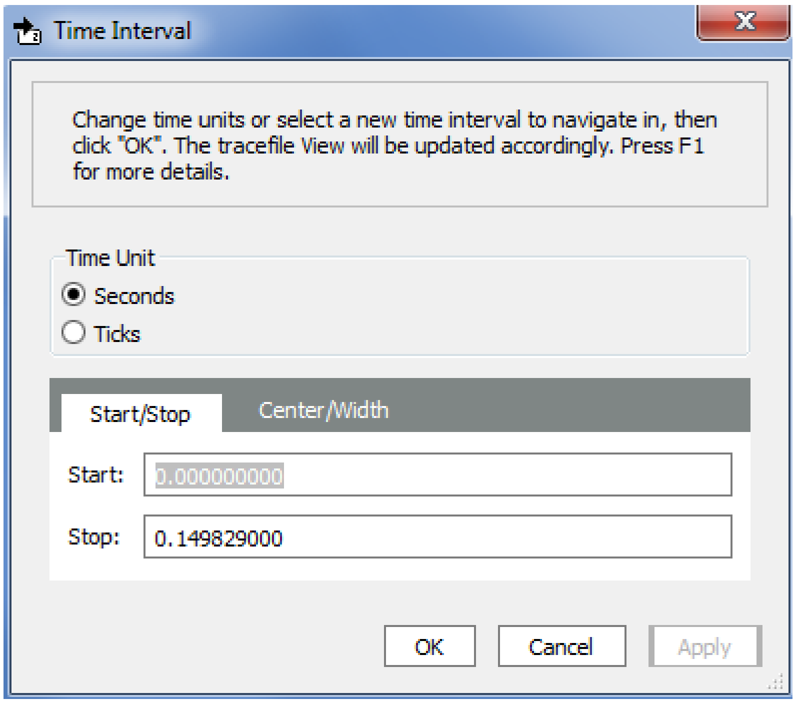
Details Dialog
Edit Configuration Dialog

Time Interval Selection

Use this dialog box to to enter or edit a time interval or duration in a filter expression or to enter a new time interval for the whole View.

To open this dialog box, go to **Navigate > Time Interval... (G)** or press the button on the toolbar. When you open this dialog box this way, it enables you to enter a new time interval for the whole View. This interval is pushed onto the zoom stack and the View is updated accordingly.

You can also open the Time Interval Selection dialog box from the filter dialog boxes/ In this case the dialog box enters or edits a time interval or duration in a filter expression.



Change time units or select a new time interval to navigate in, then click "OK". The tracefile View will be updated accordingly. Press F1 for more details.

Time Unit
[✓ Seconds]
Ticks

Start: 0.000000000

Stop: 0.149829000

Specify the time interval in ticks or seconds. You can enter the interval either by giving the start and stop or by giving the center and width. It is possible to enter a value that is greater than the maximum value of the trace file's time interval; in this case, this value is automatically reduced to the maximum value (tmax of the trace).

See Also

Zoom Stack
Filtering Dialog Box
Tagging Dialog Box

Configuration Dialog Boxes

Use these dialog boxes to manipulate the configuration that is usually stored in the `.itarc` file in your home directory upon program exit. To get the global configuration information, use the **File Default** option and to get the per file information, press the respective file name.

The global information consists of the recent file list and the list of search paths that find the source code files. The latter is stored in the **File Default/UserDefines SCLSearchPaths** option and contains a list of directories, separated by semicolon `;`. These directories are searched in the given order for source code files to be shown in the Source View dialog box.

The per-file information holds all user-defined process groups, function groups and function group colors.

See Also

Source View Dialog Box

Load Configuration File Dialog

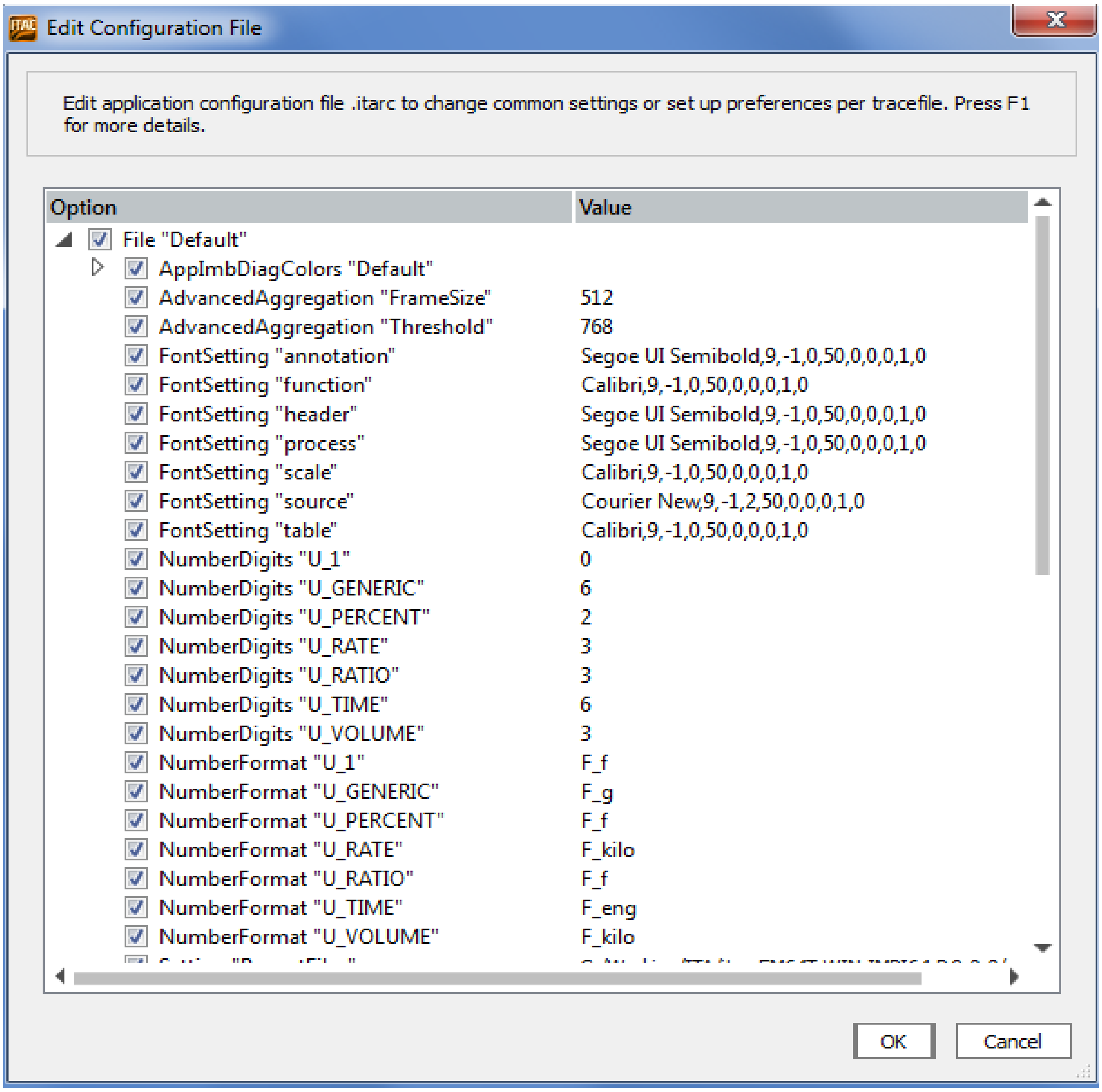
Use this dialog box to read file configuration data in the program. You can choose the information to be loaded on a file-per-file basis. You cannot save a configuration in the Intel® Trace Analyzer directly, but you can save portions of the `.itarc` file using a text editor and saving it then in the usual manner.

Use the **Merge** check box with the current configuration to choose if the configuration that is present in memory for chosen files should be replaced by or merged with the configuration from the selected file.

Edit Configuration File Dialog

Use this dialog box to edit the configuration file. For example, you can change values or remove entries from the configuration.

Use this dialog box together with the **Load Configuration File** dialog box and store per-trace file settings together with the trace file to share the definitions of process groups and function groups in a work group.


Edit Configuration File

Edit application configuration file .itarc to change common settings or set up preferences per tracefile. Press F1 for more details.

Option	Value
◢ ✓ File "Default"	
▷ ✓ ApplmbDiagColors "Default"	
✓ AdvancedAggregation "FrameSize"	512
✓ AdvancedAggregation "Threshold"	768
✓ FontSetting "annotation"	Segoe UI Semibold,9,-1,0,50,0,0,0,1,0
✓ FontSetting "function"	Calibri,9,-1,0,50,0,0,0,1,0
✓ FontSetting "header"	Segoe UI Semibold,9,-1,0,50,0,0,0,1,0
✓ FontSetting "process"	Segoe UI Semibold,9,-1,0,50,0,0,0,1,0
✓ FontSetting "scale"	Calibri,9,-1,0,50,0,0,0,1,0
✓ FontSetting "source"	Courier New,9,-1,2,50,0,0,0,1,0
✓ FontSetting "table"	Calibri,9,-1,0,50,0,0,0,1,0
✓ NumberDigits "U_1"	0
✓ NumberDigits "U_GENERIC"	6
✓ NumberDigits "U_PERCENT"	2
✓ NumberDigits "U_RATE"	3
✓ NumberDigits "U_RATIO"	3
✓ NumberDigits "U_TIME"	6
✓ NumberDigits "U_VOLUME"	3
✓ NumberFormat "U_1"	F_f
✓ NumberFormat "U_GENERIC"	F_g
✓ NumberFormat "U_PERCENT"	F_f
✓ NumberFormat "U_RATE"	F_kilo
✓ NumberFormat "U_RATIO"	F_f
✓ NumberFormat "U_TIME"	F_eng
✓ NumberFormat "U_VOLUME"	F_kilo

The entries are shown with check box like handles that are all checked by default.

To remove the entry from current configuration, uncheck it and press **OK**. To edit a value, double-click on the respective entry.

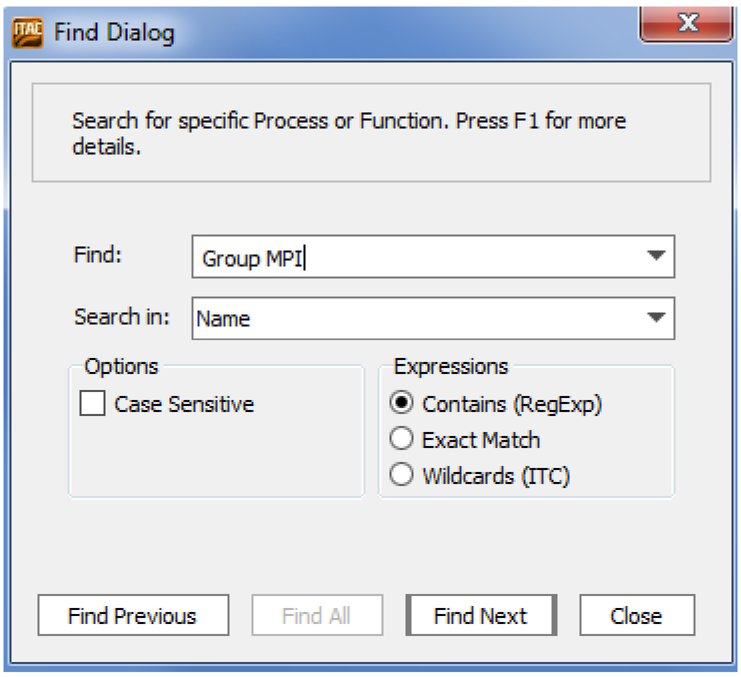
You can drag and drop group definitions and color assignments from one trace file branch to another or even onto the branch **File Default** and make them available for each new trace file. The dialog box enables you to drag and drop every child node in the configuration tree, and thus the configuration manipulations can lead to surprising results. In such cases, start again with a well-known configuration.

Generally, it is better to avoid moving options whose values are files or lists of files. For example, a cache file usually corresponds to a particular trace file, and moving a **UserDefines Cachefile** to another trace file or the default section would not make sense, unless the intent is to reuse that cache file with a different trace. On the other hand, you can drag and drop color definitions from trace to trace or to default (the latter will define that color for all trace files, unless they provide their own definition). Moving group definitions, however, requires more attention, since group ids in one trace might not make sense in another. To solve such issues, you can examine and edit values.

Find Dialog Box

Use the **Find** dialog box to search for a process or a function.

Open this dialog box through the context menu of the Function Profile and in the context menus of the Function Aggregation and the Process Aggregation dialog boxes.


Find Dialog

Search for specific Process or Function. Press F1 for more details.

Find: Group MPI

Search in: Name

Options
☐ Case Sensitive

Expressions
● Contains (RegExp)
○ Exact Match
○ Wildcards (ITC)

Use the **Search in** drop-down menu, to identify the search parameter: Name, All Columns or Dynamic Path.

Check the **Case Sensitive** box, to make search case-sensitive.

You can optimize the search process selecting the necessary type of expression. The expressions are:

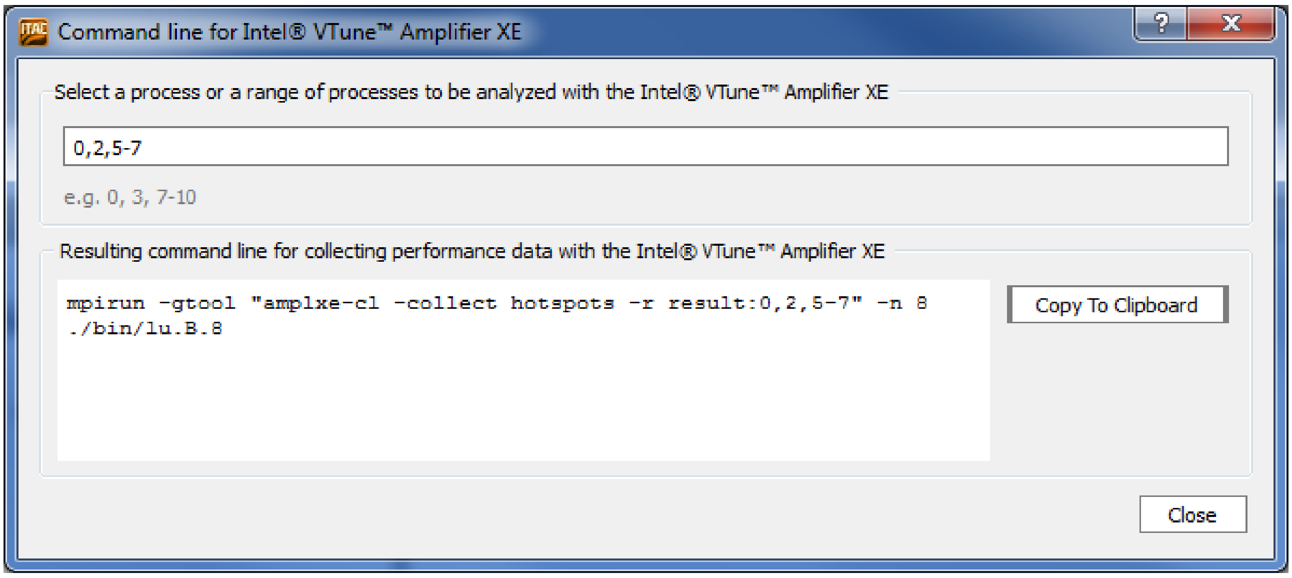
Expression:	**Description:**
Contains (RegExp)	This entry searches for a phrase that contains a regular expression. A regular expression (RegExp) provides a way to find patterns within text. Regular expressions are built up from expressions, quantifiers and assertions. The simplest form of an expression is a character, like **x** or **5**. An expression can also be a set of characters. For example, [ABCD] would match an **A** or a **B** or a **C** or a **D**. You can get more information on the valid regular expressions at http://qt-project.org/doc/qt-4.8/qregexp.html#details.

Exact Match	This entry searches for the exact match of a given character or set of strings.
Wildcards (ITC)	This option defines a search for any function that matches the given pattern. Syntax and semantics are the same as in the regular expressions used in the Intel® Trace Collector. The wildcard characters in use are *****, ******, **?** and **[]**. They match any number of characters except for the colon **:**. Pattern matching is not case-sensitive. The state or function name that the pattern is applied to consists of a class name and the symbol name, separated by a colon. The colon is special and is not matched by the ***** or **?** wildcard. To match it use ******. Examples of valid Wild Card patterns are: `MPI:*` (all MPI functions) `**:*send*` (any function that contains "send" inside any class) `MPI:*send*` (only send functions in MPI)

Command line for Intel® VTune™ Amplifier XE and Intel® Advisor XE Dialog Box

Use this dialog box to create a command line for Intel® VTune™ Amplifier XE or Intel® Advisor XE. See the Interoperability with Intel® VTune™ Amplifier XE and Intel® Advisor XE section for details.

An example dialog box for Intel VTune Amplifier may look as follows:


Command line for Intel® VTune™ Amplifier XE
Select a process or a range of processes to be analyzed with the Intel® VTune™ Amplifier XE
Resulting command line for collecting performance data with the Intel® VTune™ Amplifier XE
Copy To Clipboard
Close

0,2,5-7

e.g. 0, 3, 7-10

```
mpirun -gtool "amplxe-cl -collect hotspots -r result:0,2,5-7" -n 8
./bin/lu.B.8
```

Select This:	**To Do This:**
Process editing field	Type in the numbers of processes you want to analyze with Intel VTune Amplifier/Intel Advisor. You can select single processes separated by comma, or a range of processes separated by a minus sign. For example, to analyze processes 0, 2, 5, 6 and 7 you can type: `0,2,5-7` For full description of the syntax for `-gtool`, see the Intel® MPI Library documentation.
Resulting command line field	Contains the resulting command line, which you can modify according to your needs and use for analyzing the specified

	processes. For details on the options you can use, see the Intel VTune Amplifier/Intel Advisor documentation. When editing this field, the process editing field is unavailable. If your trace file does not contain the original command line, you will see only an example command line.

See Also

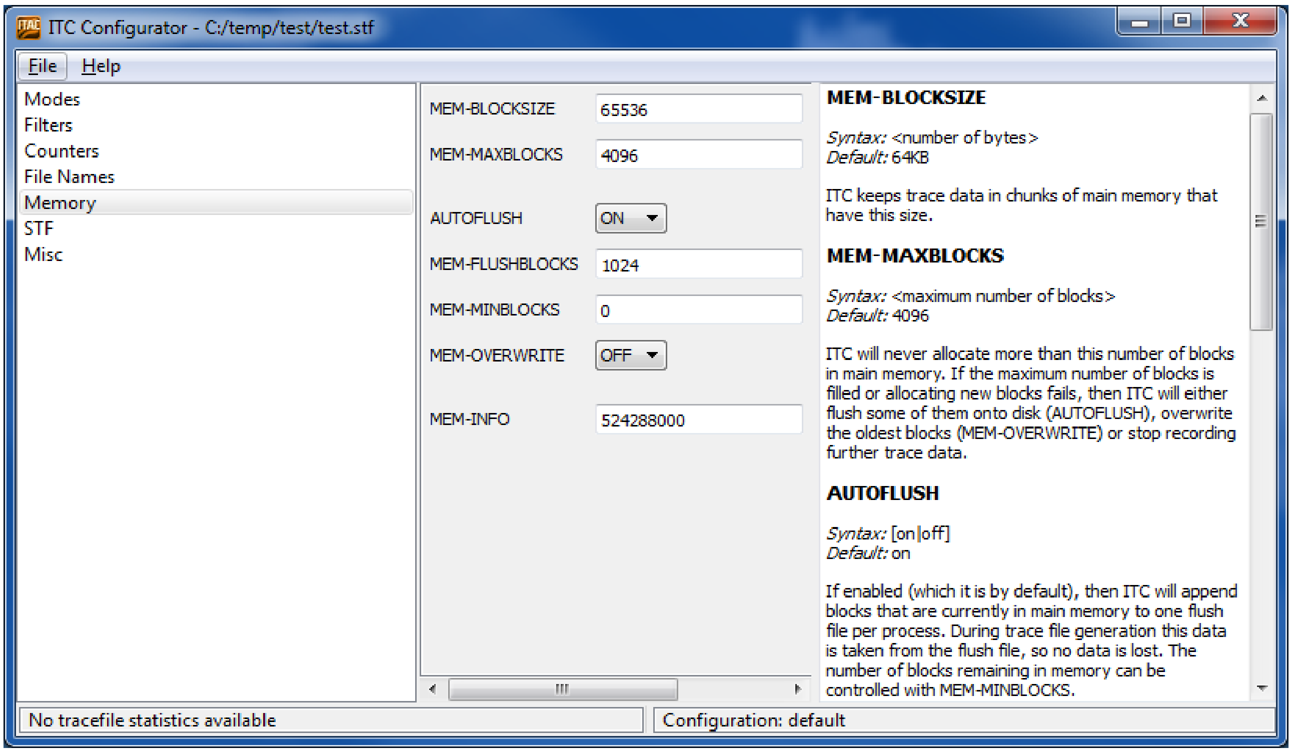
Interoperability with Intel® VTune™ Amplifier XE and Intel® Advisor XE

Configuration Assistant

Intel® Trace Collector Configuration Assistant provides a simple graphical user interface to view and edit the Intel Trace Collector configuration files that are used to define various filters for trace data collection. It can also estimate an average size of the trace file taking into account the modified parameters.

To access the Configuration Assistant, select **Views > Configure Trace Collector**.

Configuration Assistant GUI



MEM-BLOCKSIZE 65536
MEM-MAXBLOCKS 4096
AUTOFLUSH ON
MEM-FLUSHBLOCKS 1024
MEM-MINBLOCKS 0
MEM-OVERWRITE OFF
MEM-INFO 524288000

MEM-BLOCKSIZE

Syntax: <number of bytes>
Default: 64KB

ITC keeps trace data in chunks of main memory that have this size.

MEM-MAXBLOCKS

Syntax: <maximum number of blocks>
Default: 4096

ITC will never allocate more than this number of blocks in main memory. If the maximum number of blocks is filled or allocating new blocks fails, then ITC will either flush some of them onto disk (AUTOFLUSH), overwrite the oldest blocks (MEM-OVERWRITE) or stop recording further trace data.

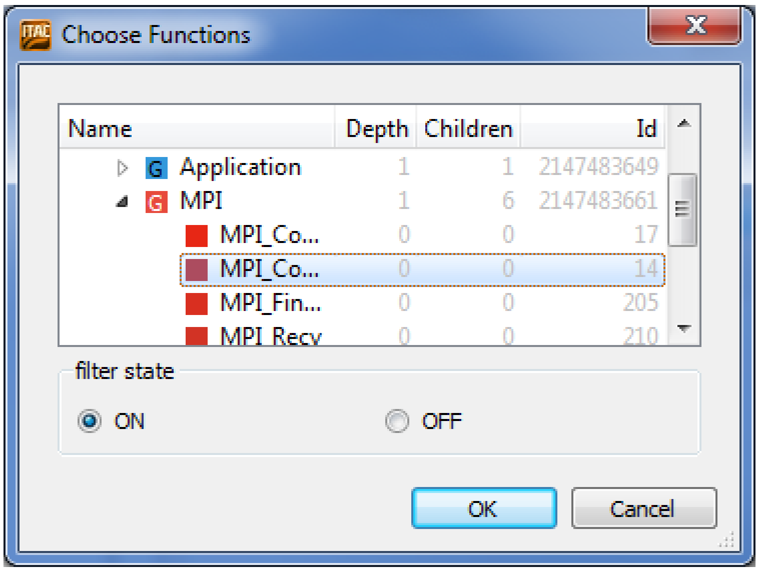
AUTOFLUSH

Syntax: [on|off]
Default: on

If enabled (which it is by default), then ITC will append blocks that are currently in main memory to one flush file per process. During trace file generation this data is taken from the flush file, so no data is lost. The number of blocks remaining in memory can be controlled with MEM-MINBLOCKS.

No tracefile statistics available
Configuration: default

Select This:	**To Do This:**
File	Create, Save and Open configuration files.
Help	Get help on the configuration parameters used on the current page.
Left column	Select from the groups of configuration parameters.
Center column	Edit the values of configuration parameters contained in the selected parameter group. For the **Filters** group you can select processes and functions by right-clicking the text editing field.

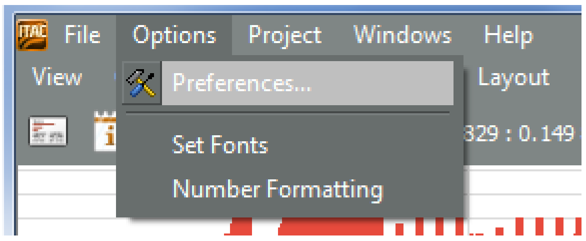
	 Choose Functions Name | Depth | Children | Id Application | 1 | 1 | 2147483649 MPI | 1 | 6 | 2147483661 MPI_Co... | 0 | 0 | 17 MPI_Co... | 0 | 0 | 14 MPI_Fin... | 0 | 0 | 205 MPI_Recv | 0 | 0 | 210 filter state: [✓ ON] OFF OK Cancel
Right column	See detailed description of each configuration parameter from the selected parameters group. For description of all configuration parameters in alphabetical order see *Intel® Trace Collector User and Reference Guide*.
Left status bar	See the estimated trace file size for the currently set configuration parameters.
Right status bar	See the configuration file name currently open. Default if the assistant is working with the default configuration.

See Also

Configuring Intel® Trace Collector section in the *Intel® Trace Collector User and Reference Guide*

2.2.8. Settings

Use settings to customize your working environment. To access the settings, on the main menu bar select **Options**.



See description of the menu entries:

- Preferences
- Font Settings
- Number Formatting Settings

Preferences

Select **Preferences** to set up your working environment.

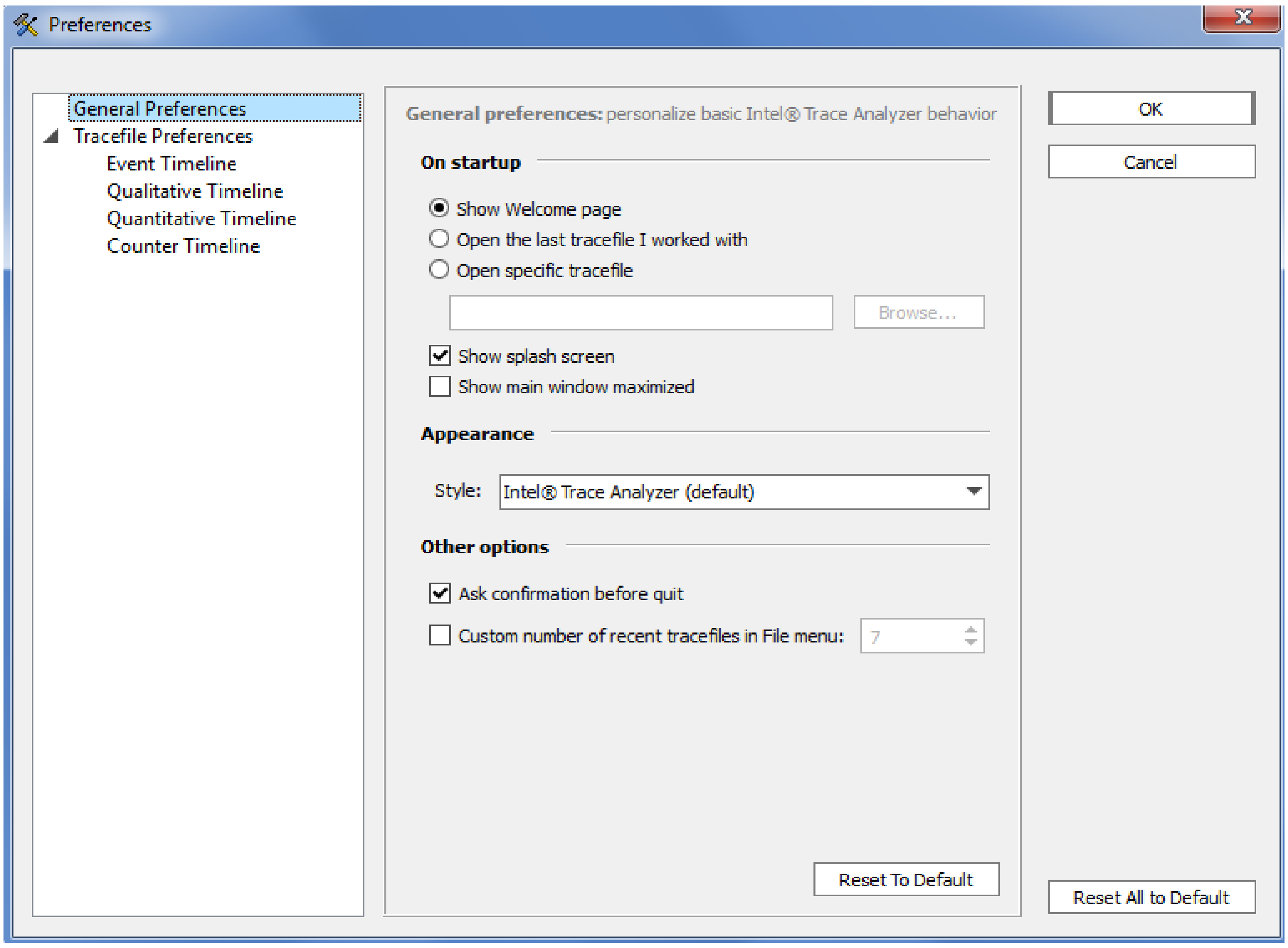
This section describes the following preference dialog boxes:

- General Preferences

- Tracefile Preferences
- Event Timeline Settings
- Qualitative Timeline Settings
- Quantitative Timeline Settings
- Counter Timeline Settings

General Preferences

The **General Preferences** enable you to set up the basic Intel® Trace Analyzer behavior, such as the startup options, view style, confirmation before quitting the Intel Trace Analyzer and the number of the recently opened tracefiles.



General preferences: personalize basic Intel® Trace Analyzer behavior

On startup

- [✓ Show Welcome page]
- Open the last tracefile I worked with
- Open specific tracefile
- [✓ Show splash screen]
- Show main window maximized

Appearance

Style: Intel® Trace Analyzer (default)

Other options

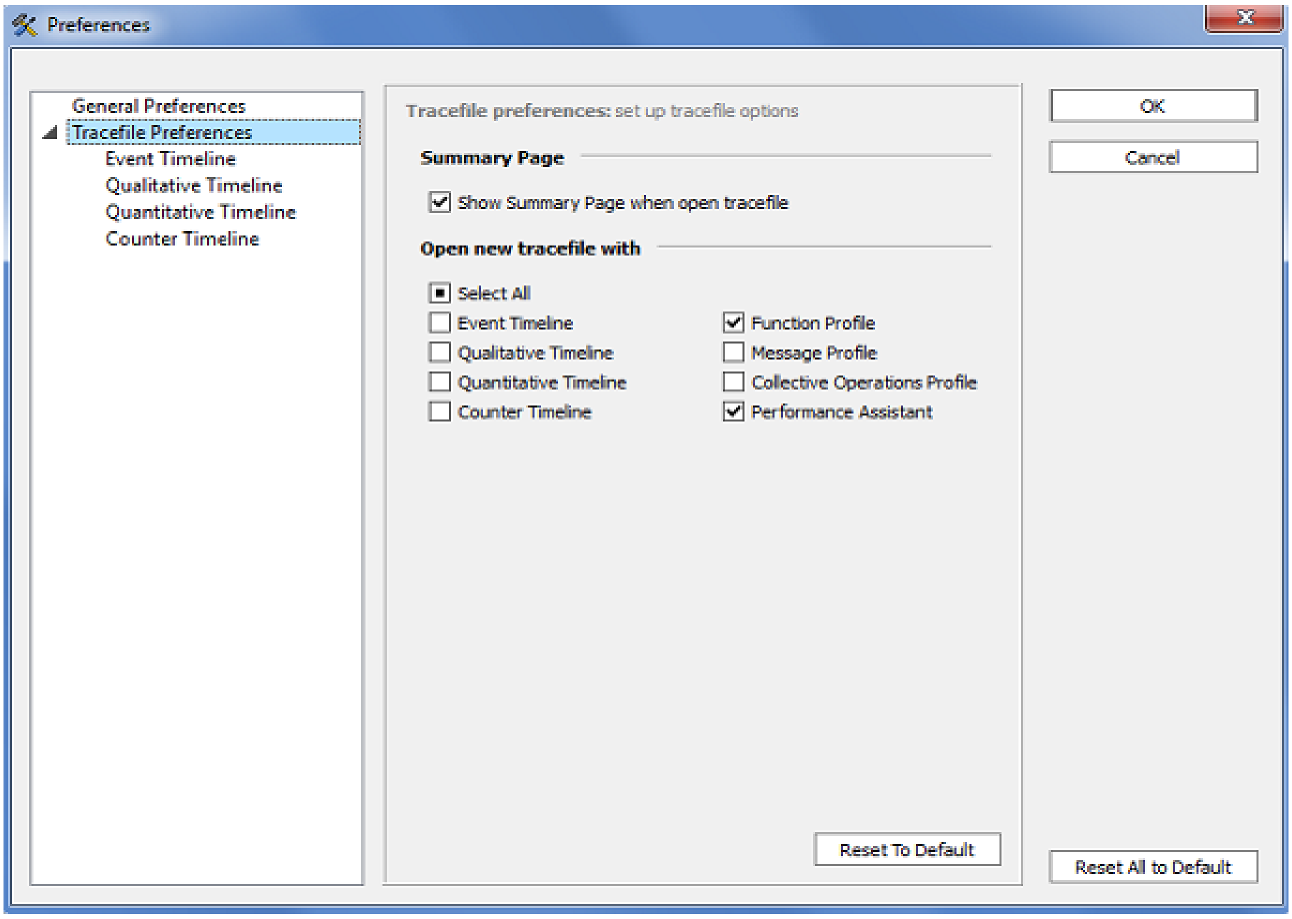
- [✓ Ask confirmation before quit]
- Custom number of recent tracefiles in File menu: 7

Configuring the Intel Trace Analyzer behavior in the **General Preferences**:

Select:	**To Do:**
On Startup	Choose the default Intel Trace Analyzer startup experience. It can be the **Welcome Page**, the last tracefile you worked with or another tracefile you have specified. In this section you can also enable/disable the **Splash Screen** appearance and the main window maximization.
Appearance	Choose the design in which the view is displayed. The options in the **Style** section depend on the environment.
Other Options	Customize the number of the recent tracefiles to appear in the **File Menu**. Enable confirmation before you quitting Intel Trace Analyzer.

Tracefile Preferences

The **Tracefile Preferences** enable you to customize which charts you want visible when opening a new tracefile or in a new view of an existing tracefile. By default, the **Summary Page** is opened. Choose Continue, to open a View with the **Function Profile** chart and **Performance Assistan**t.



Tracefile preferences: set up tracefile options

Summary Page

[✓] Show Summary Page when open tracefile

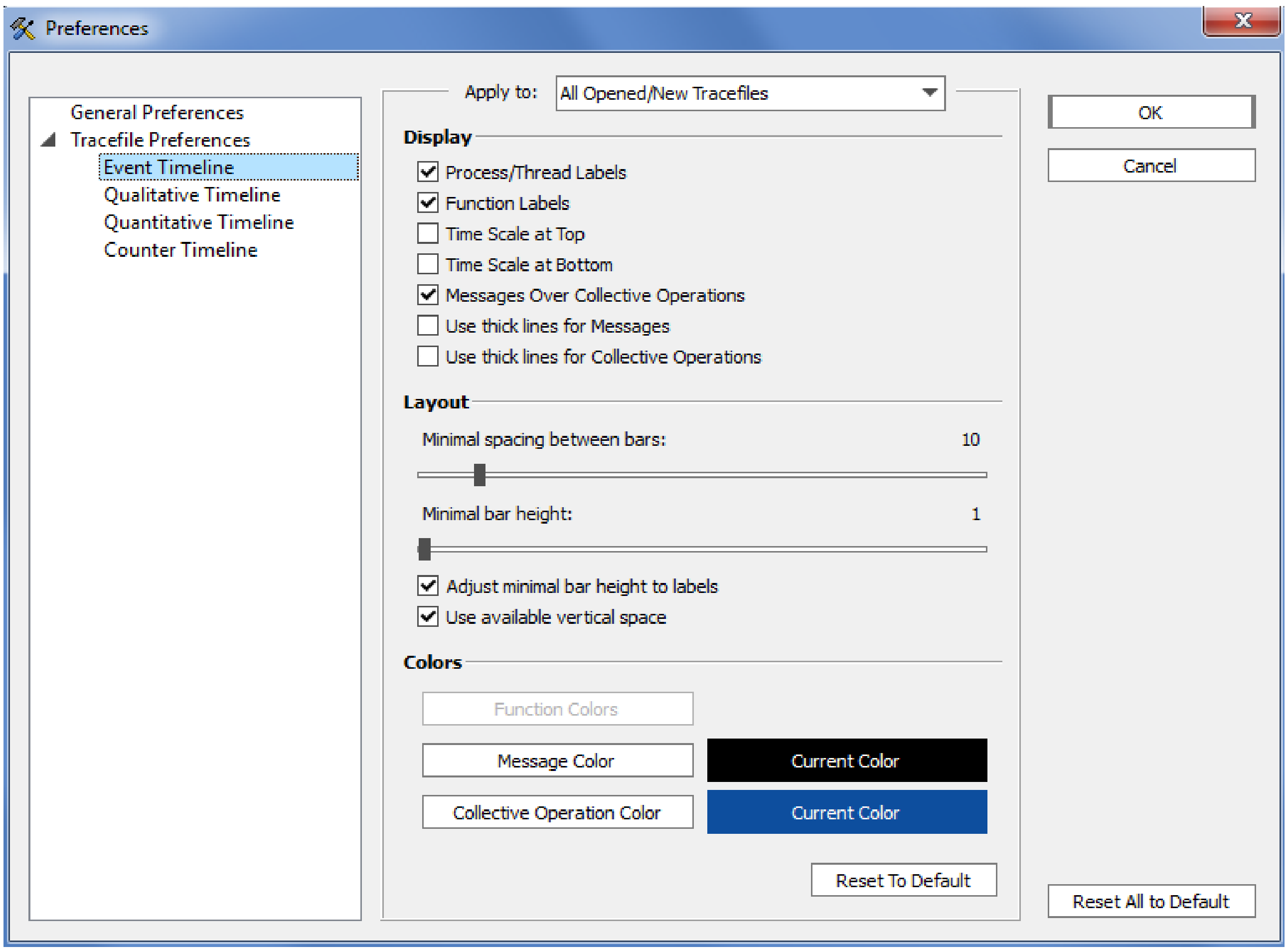
Open new tracefile with

[■] Select All

[ ] Event Timeline	[✓] Function Profile
[ ] Qualitative Timeline	[ ] Message Profile
[ ] Quantitative Timeline	[ ] Collective Operations Profile
[ ] Counter Timeline	[✓] Performance Assistant

Event Timeline Settings

Use the **Event Timeline Settings** to change the display settings, the layout settings and the colors of the Event Timeline Chart.


Preferences

- General Preferences
- Tracefile Preferences
  - Event Timeline
  - Qualitative Timeline
  - Quantitative Timeline
  - Counter Timeline

Apply to: All Opened/New Tracefiles

Display

- [✓] Process/Thread Labels
- [✓] Function Labels
- [ ] Time Scale at Top
- [ ] Time Scale at Bottom
- [✓] Messages Over Collective Operations
- [ ] Use thick lines for Messages
- [ ] Use thick lines for Collective Operations

Layout

Minimal spacing between bars: 10

Minimal bar height: 1

- [✓] Adjust minimal bar height to labels
- [✓] Use available vertical space

Colors

Function Colors

Message Color — Current Color

Collective Operation Color — Current Color

Setting	Impact
Minimal Spacing Between Bars	Adjust the space between the function bars.
Minimal Bar Height slider	Adjust the height of the function bar itself.
Adjust Minimal Bar Height to Labels	Make the bars tall enough to display a function label. This option is checked by default.
Use Available Vertical Space	Influence the overall vertical layout of the bars. This option is checked by default.
Function Colors	Change the color used in timelines and profiles to mark functions
Message Color	Change the color of messages
Collective Operation Color	Change the color of Collective Operations

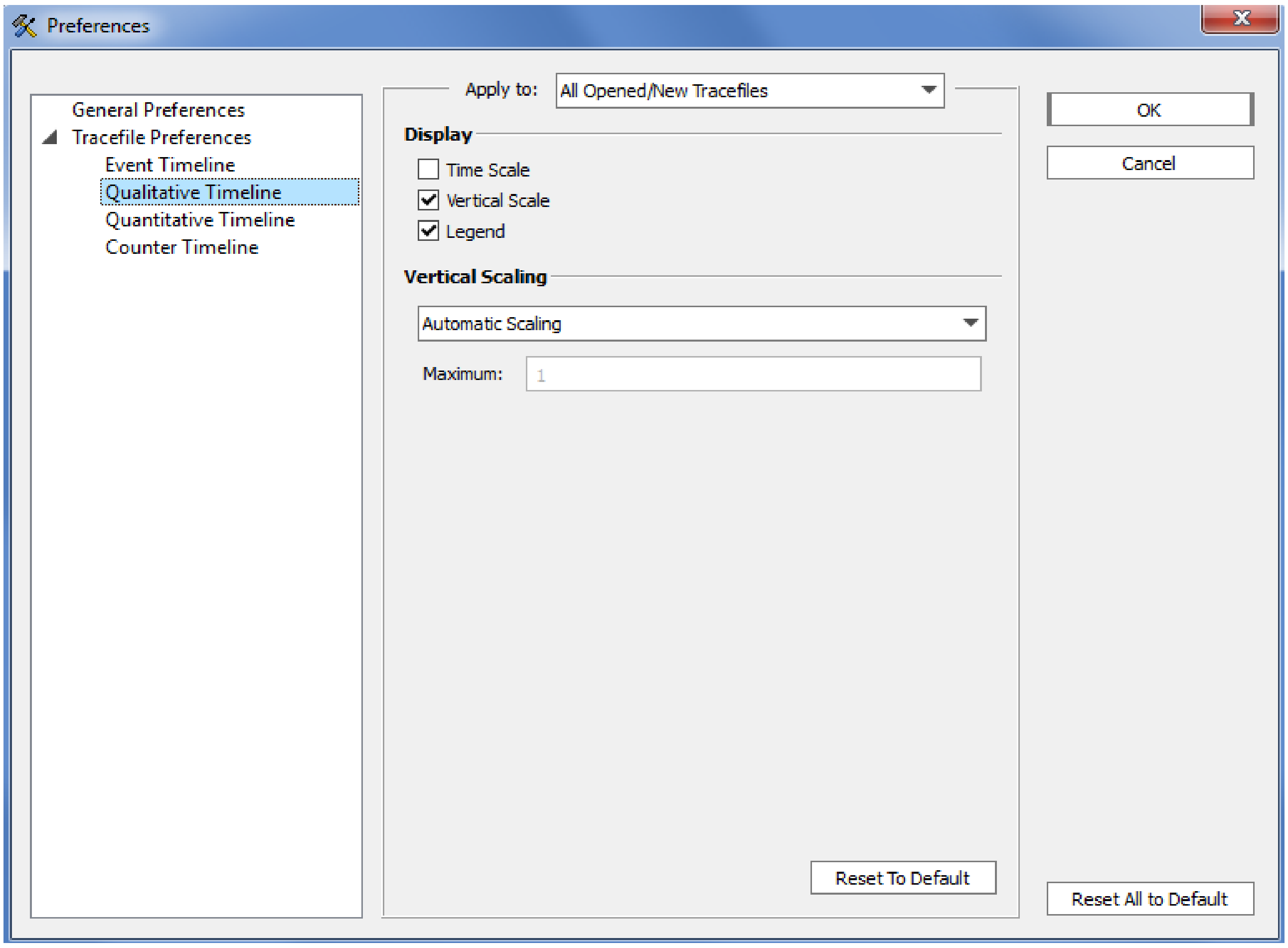
The customized colors chosen for messages and collective operations are local to the Event Timeline Chart. But the customized colors chosen for functions or function groups will be used by all Charts and Views in the same tracefile.

See Also

Function Group Color Editor

Qualitative Timeline Settings

The **Qualitative Timeline Settings** offers controls for the Display and Vertical Scaling of the timeline.



Apply to: All Opened/New Tracefiles

Display
- [ ] Time Scale
- [✓] Vertical Scale
- [✓] Legend

Vertical Scaling: Automatic Scaling

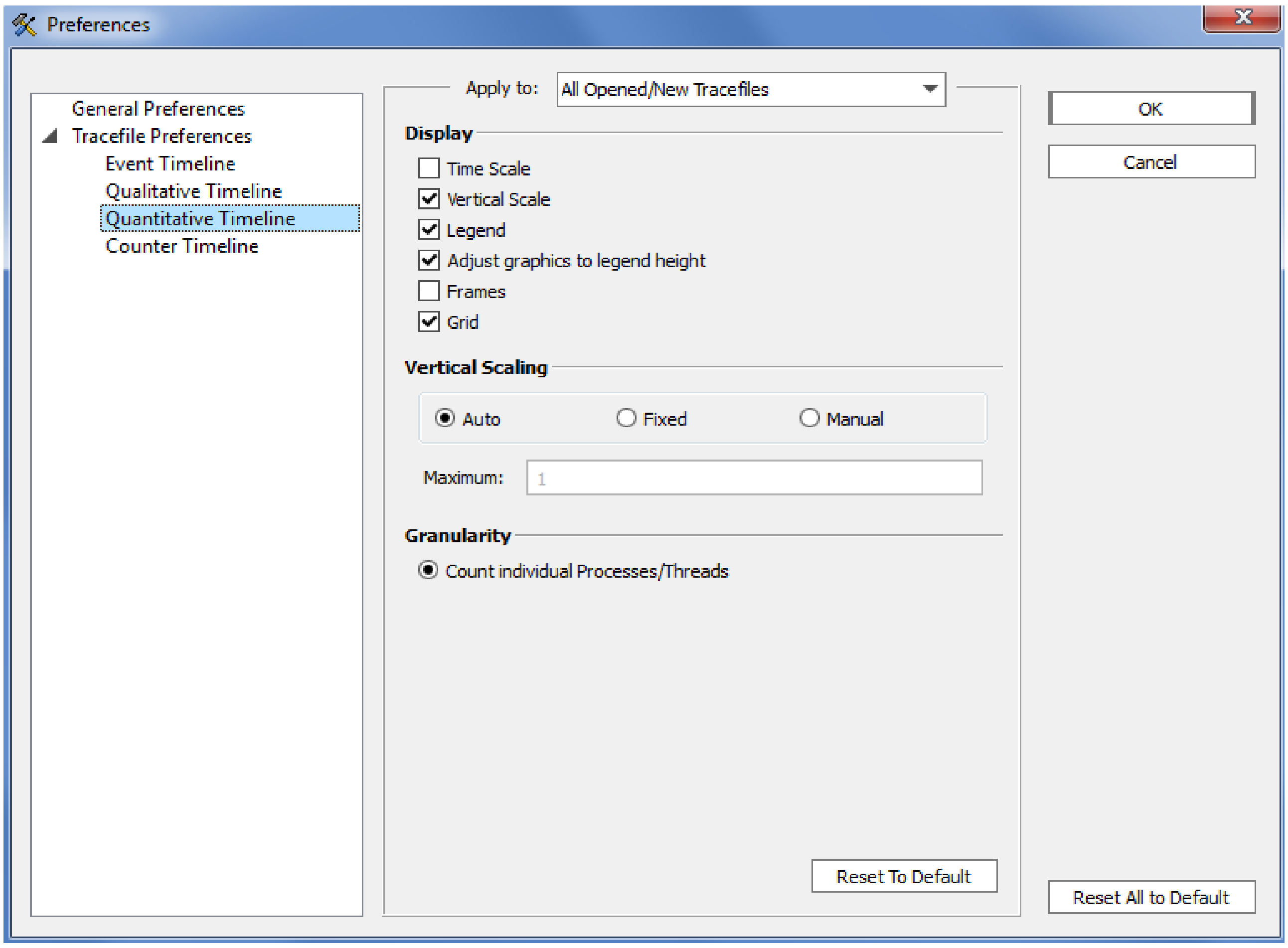
Maximum: 1

Adjust **Display** to control showing the scales and legend. By default, the vertical scale and the legend are shown.

Use the **Vertical Scaling** to switch between the default **Automatic Scaling** and the **Manual Scaling** of the y-axis. To specify explicitly the maximum scale value, use **Manual Scaling**. To compare visually two or more Charts in the same or distinct Views , you will need to specify the same maximum value for the charts.

Quantitative Timeline Settings

Quantitative Timeline Settings control the display options and the scaling of the Qualitative Timeline Chart.



Apply to: All Opened/New Tracefiles

Display

- [ ] Time Scale
- [✓] Vertical Scale
- [✓] Legend
- [✓] Adjust graphics to legend height
- [ ] Frames
- [✓] Grid

Vertical Scaling

[● Auto] Fixed Manual

Maximum: 1

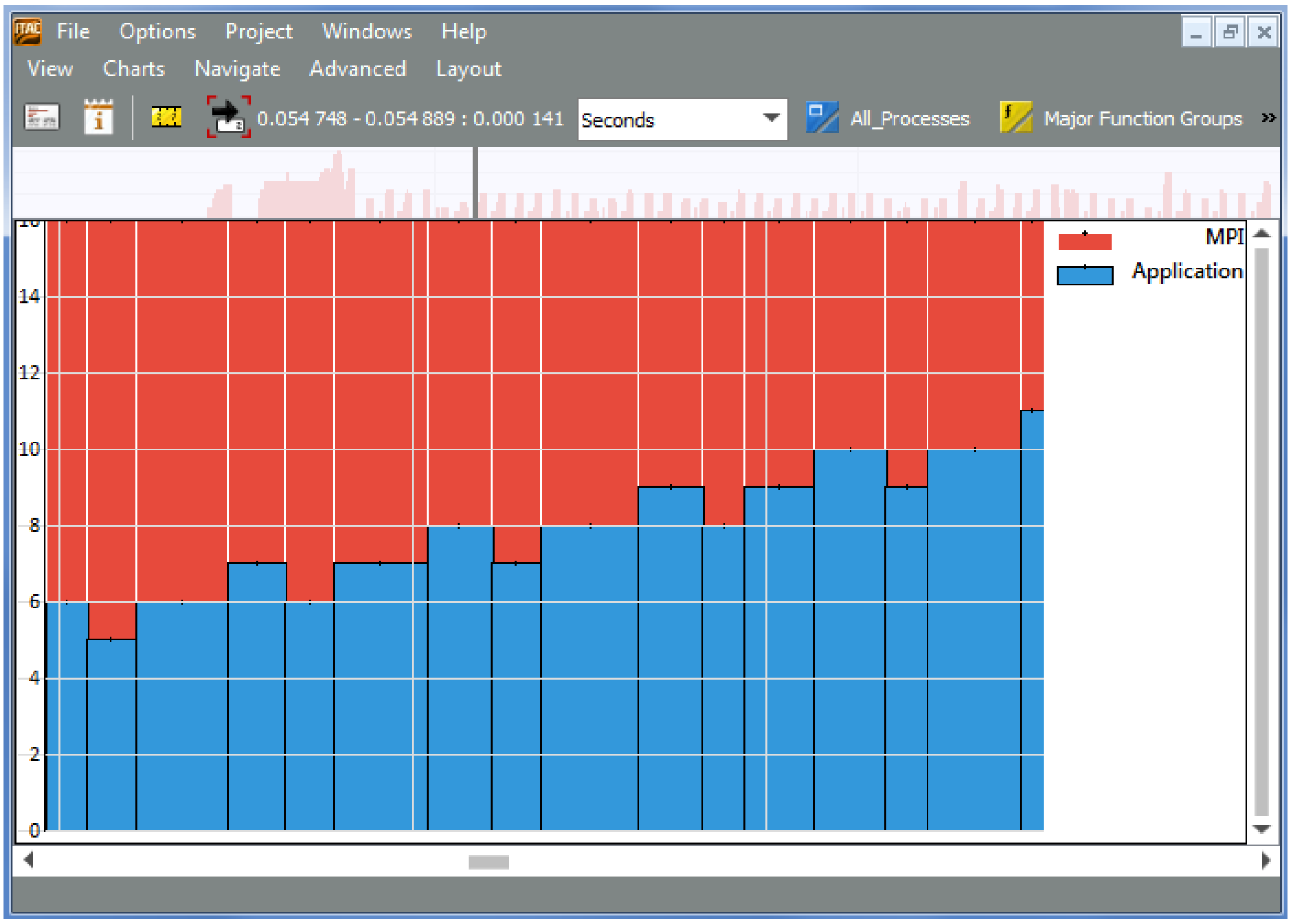
Granularity

[● Count individual Processes/Threads]

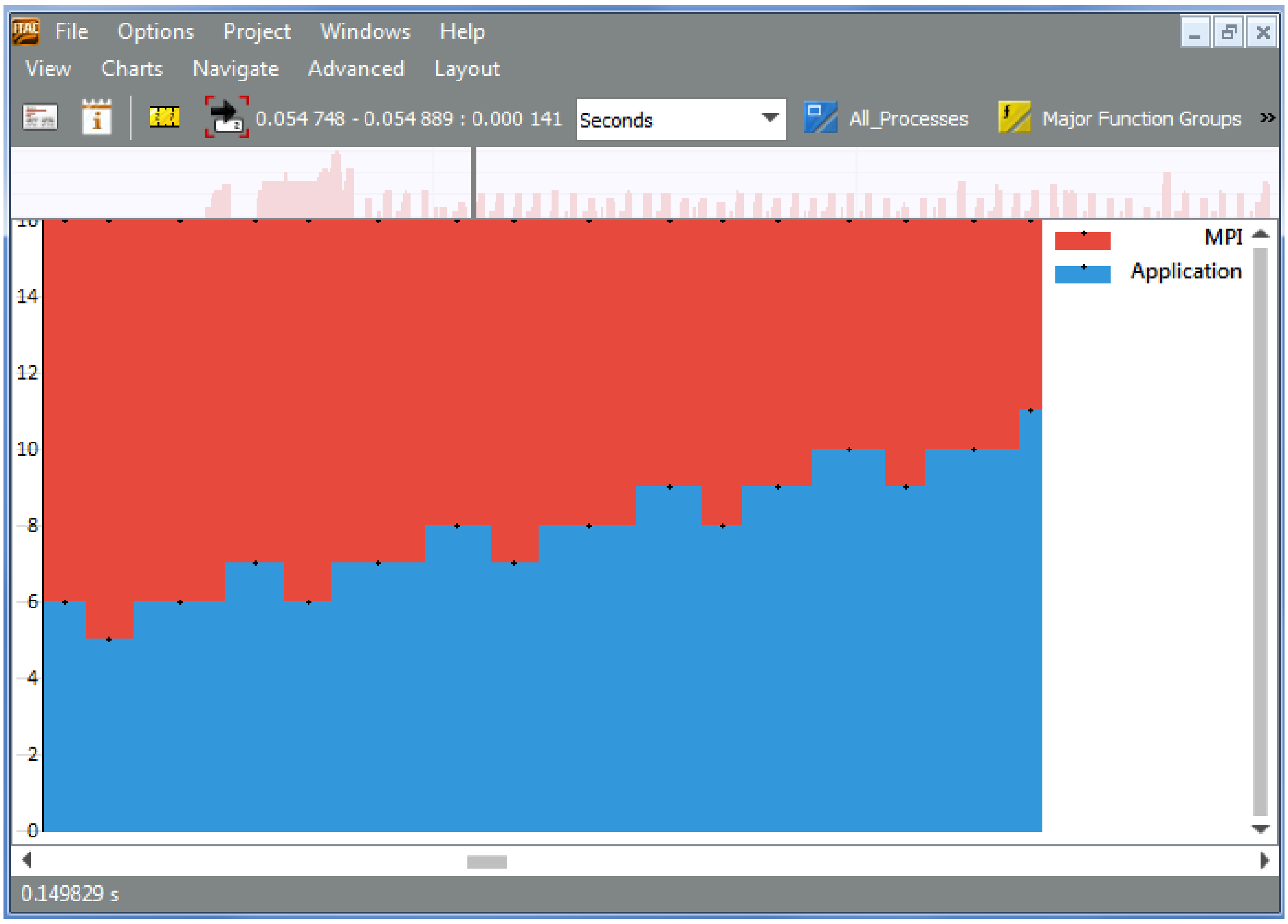
Qualitative Timeline Settings offer the following controls:

Setting	Impact
Time Scale	Display a time scale along the x-axis
Vertical Scale	Show the scale on the y-axis
Legend	Show the legend in the right margin of the timeline
Adjust graphics to legend height	Make the size of the diagram large enough to show all legend items
Frames	Frames outline the bars. The frames become visible only when you zoom in very closely so that each bar is separated from the other.
Grid	Draw the grid on top of the data and align it with the ticks on the scales
Vertical Scaling	Select the mode of scaling: either **Auto**, **Fixed**, or **Manual**. Specify the maximum scaling value in the manual mode.
Granularity	Define the level of detail you want the Quantitative Timeline to display. It may be either Individual processes/Threads or just top level entries of Group_All_Processes.

Example of using the **Frames** box to outline the bars in the Quantitative Timeline:



Example of when the **Grid** is unchecked:

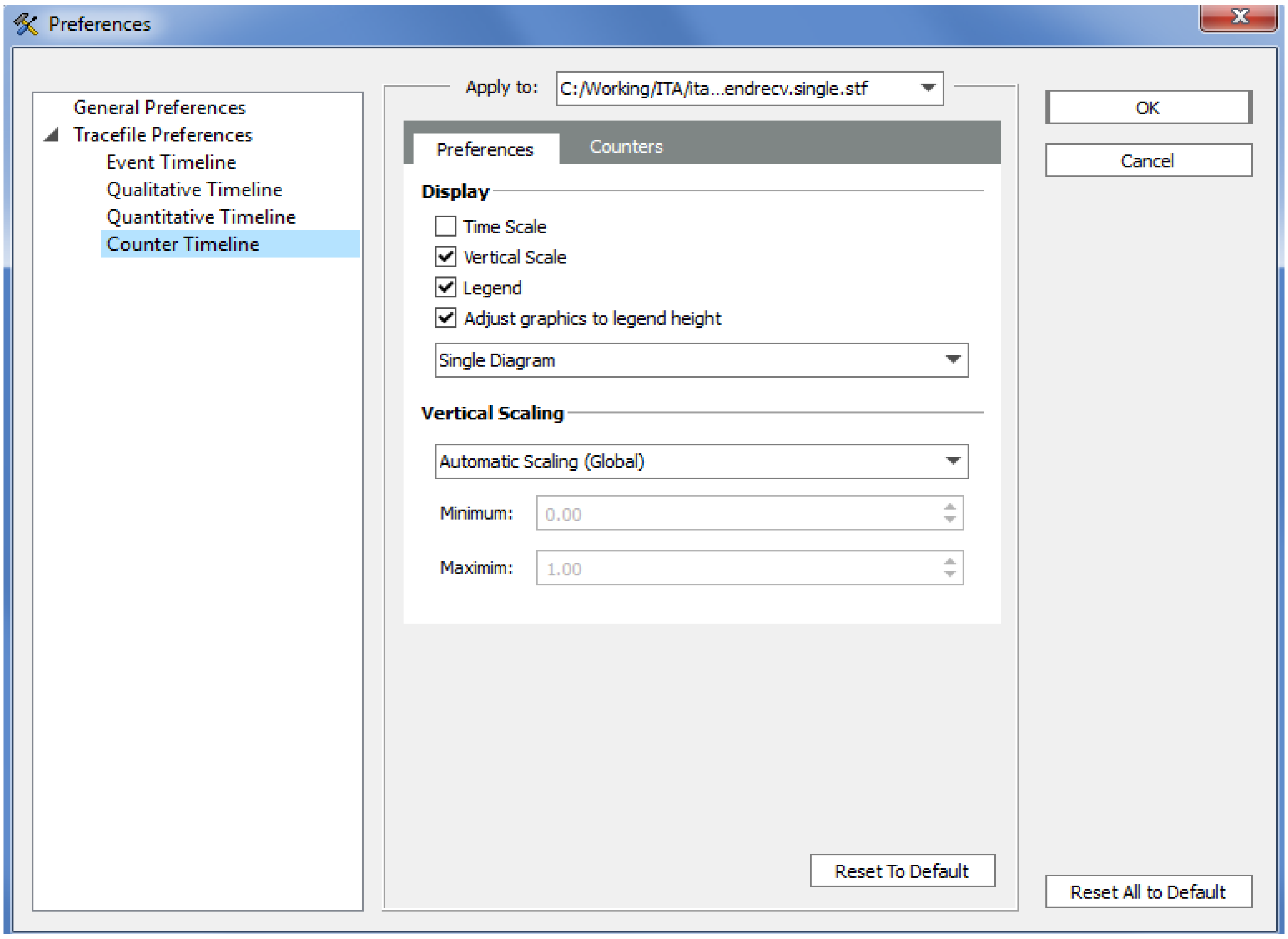


Counter Timeline Settings

The **Counter Timeline Settings** provide **Preferences** and **Counters** controls.

Preferences

Use these controls to adjust the display settings and the Counter Timeline scaling.



Apply to: C:/Working/ITA/ita...endrecv.single.stf

Display

- [ ] Time Scale
- [✓] Vertical Scale
- [✓] Legend
- [✓] Adjust graphics to legend height

Single Diagram

Vertical Scaling

Automatic Scaling (Global)

Minimum: 0.00

Maximim: 1.00

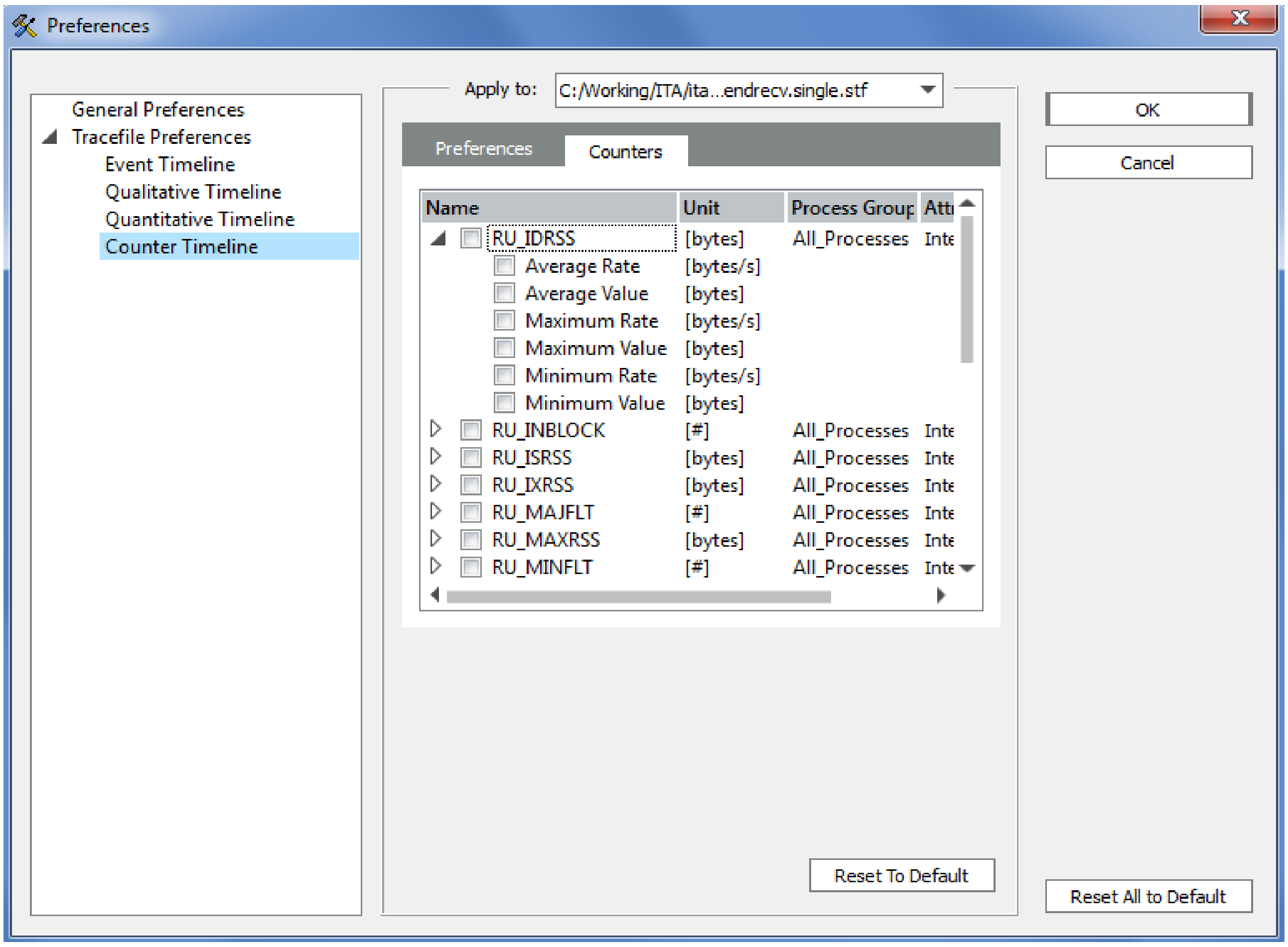
In the **Display** group, configure the Counter Timeline with the following controls:

Setting	Impact
Time Scale	Display the time scale along the x-axis
Vertical Scale	Display the time scale along the y-axis. By default, this option is enabled so that you can see the scale.
Legend	Show the legend in the right margin of the timeline. It is also enabled by default.
Adjust graphics to legend height	Make the size of the diagram large enough to show all legend items
Single Diagram/Many Diagrams combo-box	Select either a single diagram for all selected counters and target group children or a separate diagram for each target group child

Vertical Scaling offers selection between the **Automatic Scaling (per Diagram)**, **Automatic Scaling (Global)** and the **Manual Scaling** of the y-axis. To specify explicitly the minimum and maximum scale values, choose **Manual Scaling**. When visually comparing two or more Charts in the same or distinct Views, specify the same bounds for all charts.

Counters

There are seven entries for each trace file counter. The top level entry for each counter shows the counter Name, its Unit, the target Process Group, and Attributes. The name and the unit are arbitrary, free-format strings defined in the trace file.



Name	Unit	Process Group	Attı
RU_IDRSS	[bytes]	All_Processes	Inte
Average Rate	[bytes/s]		
Average Value	[bytes]		
Maximum Rate	[bytes/s]		
Maximum Value	[bytes]		
Minimum Rate	[bytes/s]		
Minimum Value	[bytes]		
RU_INBLOCK	[#]	All_Processes	Inte
RU_ISRSS	[bytes]	All_Processes	Inte
RU_IXRSS	[bytes]	All_Processes	Inte
RU_MAJFLT	[#]	All_Processes	Inte
RU_MAXRSS	[bytes]	All_Processes	Inte
RU_MINFLT	[#]	All_Processes	Inte

The column **Process Group** contains the counter target group. A counter that has different values for each process has the target group All_Processes, while a counter that has a distinct value for each SMP node belongs to the target group All_Nodes.

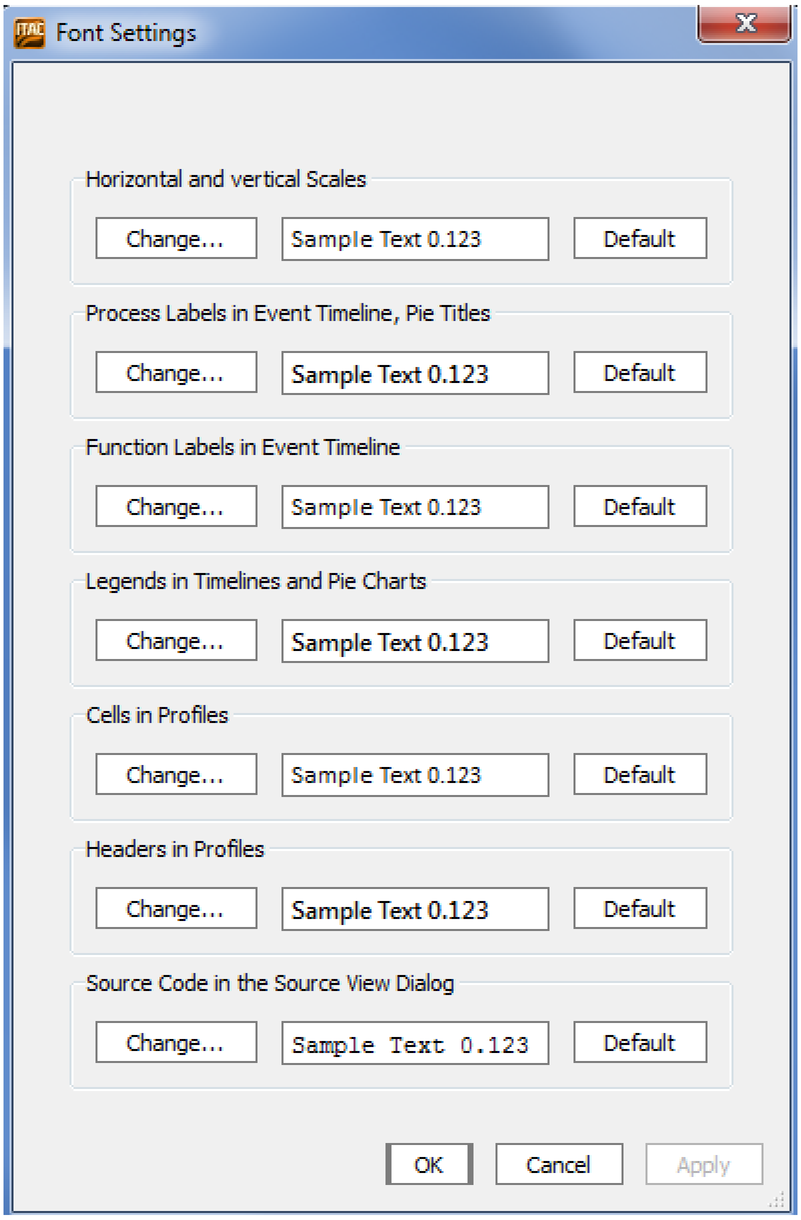
The column **Attributes** can contain either the Integer or Double attribute, indicating the counter type. And **Valid BEFORE Point**, **Valid AFTER Point**, **Valid AT Point** or **Curve** attributes to indicate the counter scope. For example, curve indicates that it is meaningful to interpolate values between two given counter values. The attribute **Show Rate** indicates that it is preferable to display the derivation to time instead of the plain counter values.

Under the top-level counter entry, there are six entries for the minimum, average and maximum values and for the minimum, average and maximum rates, which you can switch on and off independently. If you just check the top-level box with the name of the counter, either the average value or average rate is chosen depending on the counter attributes.

Font Settings

To change the fonts in any part of Intel Trace Analyzer, navigate to **Options > Set Fonts**.

In the **Fonts** dialog box, you can change the fonts of the labels, legends or headers in the different Charts, for instance. To modify a particular font, click the **Change** button and change the size, style and font in the opened dialog box and select **OK** to preserve your selection.



Font Settings

Horizontal and vertical Scales
Change... | Sample Text 0.123 | Default

Process Labels in Event Timeline, Pie Titles
Change... | Sample Text 0.123 | Default

Function Labels in Event Timeline
Change... | Sample Text 0.123 | Default

Legends in Timelines and Pie Charts
Change... | Sample Text 0.123 | Default

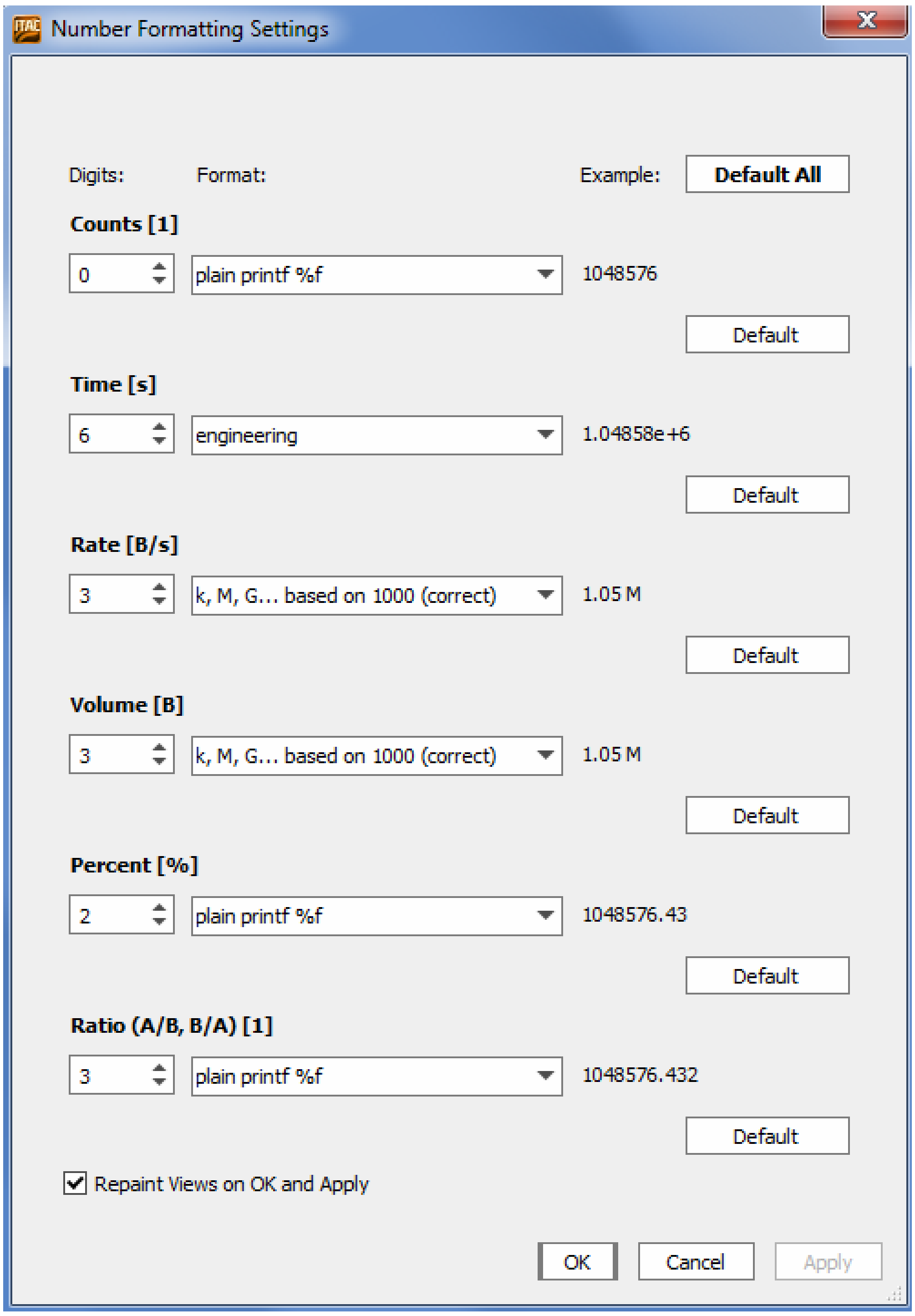
Cells in Profiles
Change... | Sample Text 0.123 | Default

Headers in Profiles
Change... | Sample Text 0.123 | Default

Source Code in the Source View Dialog
Change... | Sample Text 0.123 | Default

Number Formatting Settings

Use **Number Formatting Settings** to change the way numerical values of different units are represented. These settings enable you to change the number of digits shown and the format, in which the numerical value is shown for each unit. The number of digits is either interpreted as digits after the decimal point or as the number of valid (non zero) digits. The exact interpretation is dependent on the chosen format bases on the plain printf-format %f or %g.



Number Formatting Settings

Digits:	Format:	Example:
Counts [1]		
0	plain printf %f	1048576
Time [s]		
6	engineering	1.04858e+6
Rate [B/s]		
3	k, M, G... based on 1000 (correct)	1.05 M
Volume [B]		
3	k, M, G... based on 1000 (correct)	1.05 M
Percent [%]		
2	plain printf %f	1048576.43
Ratio (A/B, B/A) [1]		
3	plain printf %f	1048576.432

[✓] Repaint Views on OK and Apply

You can control repainting of views. When enabled, it ensures that all open Views are repainted upon **Apply** or **OK** to take on the new values. When disabled, the changes in the format become visible only with the next update. All values can be restored to their default by using the **Default All** button or by using the individual **Default** buttons for each number type.

2.3. Navigating Timelines

In addition to the menu entries and shortcuts, there are two more ways to navigate the time interval in Intel® Trace Analyzer:

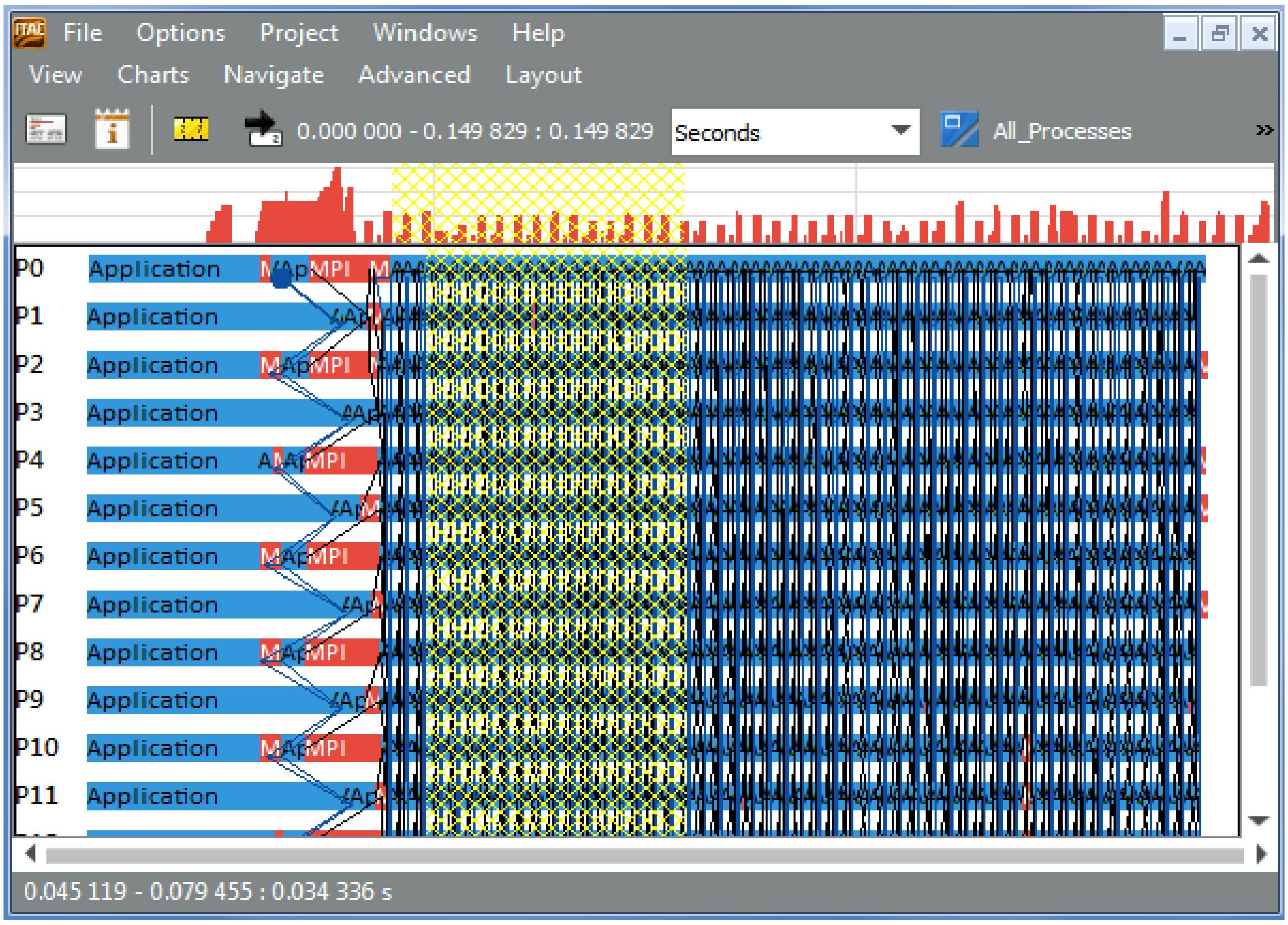
- Use the scroll bar found below the timelines
- Use the mouse to zoom into a time interval

Similar to the keystrokes, these operations manipulate the zoom stack.

To move the View by a quarter of a screen to a selected direction, click the arrow buttons of the scroll bar. To move the View by one entire screen, click into the bar. Both actions push the new interval onto the zoom stack.

To use the mouse for zooming, left-click the timeline (even clicking into the Time Scale works) and select the region of the new time interval, keeping the left mouse button pressed. On releasing the left mouse button this new interval is pushed onto the zoom stack.

Zooming into the Event Timeline:



See Also

Navigate Menu

2.3.1. Zoom Stack

The zoom stack supports navigation in time by storing previously displayed time intervals. You can restore these time intervals when it is necessary. Keyboard or mouse navigation is quite intuitive without detailed knowledge of the zoom stack. Nevertheless, an explanation is given below for the sake of complete reference.

Consider a trace file spanning the time from 0 to 100 seconds. When the first View is opened the zoom stack looks like this:

```
0:(0;100)(50;100)
```

Each stack entry has the syntax (<*start*>;<*stop*>)(<*center*>;<*width*>). Zooming in using **Zoom In** (I), magnifies the Chart by a factor of 2 around the current center. In the above example, the center is at 50 seconds. Therefore, zooming in twice using the `I` shortcut key will result in the following stack. You can see that a stack level is created for each action performed. On pressing `Ctrl+Right` twice, the time frame of the Chart is moved to the right by two window sizes:

State of the Zoom Stack - Zoomed In Twice:

```
3:(55;65)(60;10)
```

```
2:(50;70)(60;20)
1:(40;80)(60;40)
0:(0;100)(50;100)
```

For example, here is the original state of the Zoom Stack:

State of the Zoom Stack - Moved Two Window Sizes to the Right:

```

5:(75;85)(80;10)

4:(65;75)(70;10)
3:(55;65)(60;10)
2:(50;70)(60;20)
1:(40;80)(60;40)
0:(0;100)(50;100)
```

Taking this Zoom Stack as an example, see the differences between **Back** (B), **Zoom Out** (O) and **Zoom Up** (U) explained below.

Back (B): When you press B in the original state, one level of the stack pops out and the previous time frame is displayed:

State of the Zoom Stack after *Back (B)*:

```
4:(65;75)(70;10)
3:(55;65)(60;10)
2:(50;70)(60;20)
1:(40;80)(60;40)
0:(0;100)(50;100)
```

Zoom Out (O): Using O in the original state of the Zoom Stack lowers the magnification (doubles the width) and pushes a new item on the stack. Zooming out using O in the original state results in the stack as shown below:

State of the Zoom Stack after *Zoom Out (O)*:

```
6:(70;90)(80;20)
5:(75;85)(80;10)
4:(65;75)(70;10)
3:(55;65)(60;10)4
2:(50;70)(60;20)
1:(40;80)(60;40)
0:(0;100)(50;100)
```

Zoom Up (U): The resultant zoom after using U is a little more complicated. Pressing U keeps the current center and uses the magnification from before the latest zoom in (which is the change from level 2 to 3 in the original state) and to clear the stack up to and including the last zoom in found. In this example the current center is 80 and the width from stack level 2, which is 20, is used:

State of the Zoom Stack after *Zoom Up (U)*:

```
2:(70;90)(80;20)
1:(40;80)(60;40)
0:(0;100)(50;100)
```

Reset (R): Reset clears the stack and pushes one entry covering the full time range of the trace file.

2.4. Viewing Correctness Checking Reports

The Correctness Checking Reports (CCRs) are shown on three charts:

- Event Timeline CCRs
- Qualitative Timeline CCRs
- Detailed Dialog CCRs

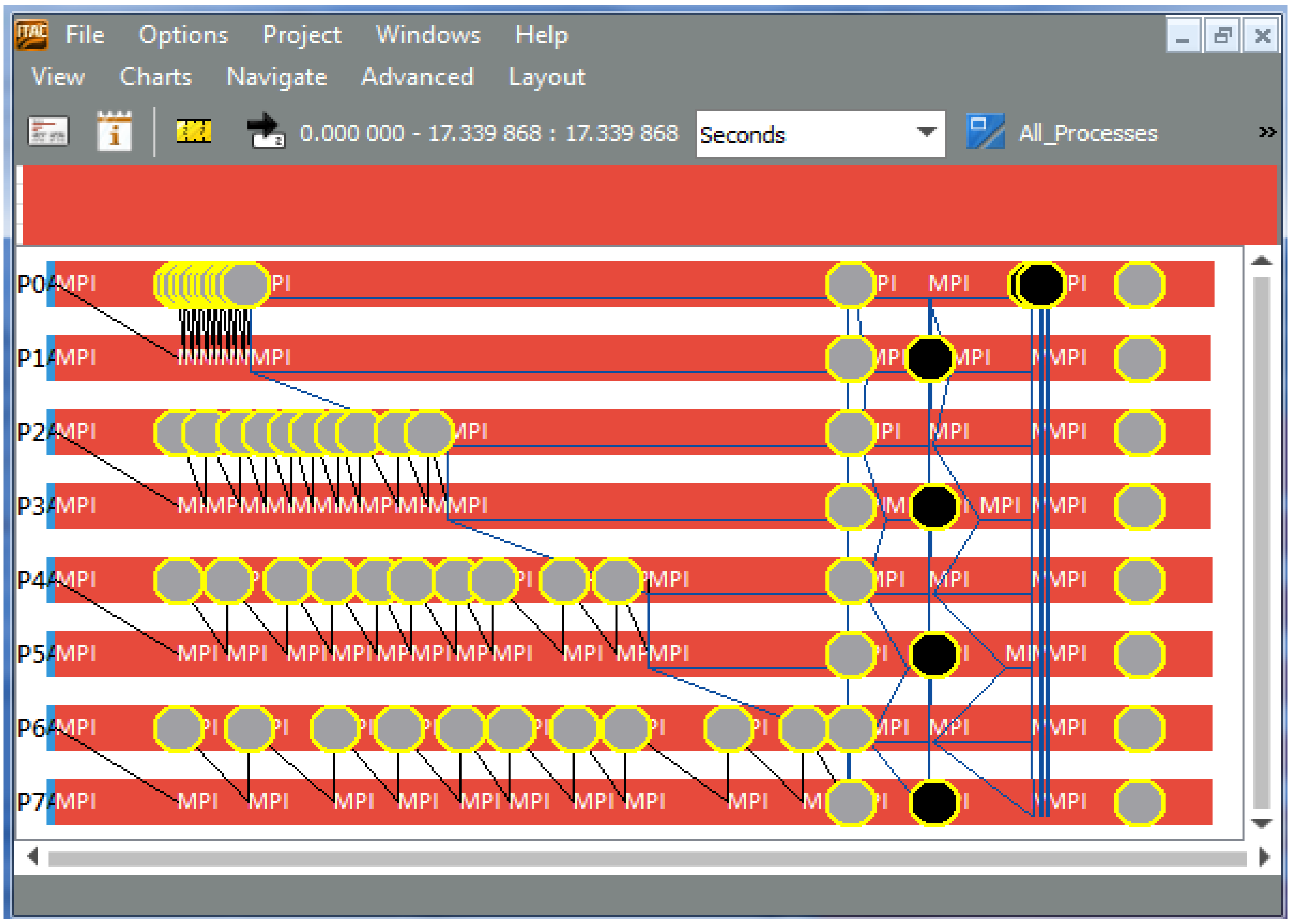
On the first two charts you can get a detailed dialog through the context menu. The detailed dialog also contains details on the CCRs.

When it is necessary, each correctness checking report (CCR) is cached like other events (Function, Messages, Collective Operations, etc). All CCRs are put into each level of the cache. The CCRs are not crowded so each cache level contains the same information. It may lead to extra memory usage but this is unlikely since the total number of CCRs is not expected to be large.

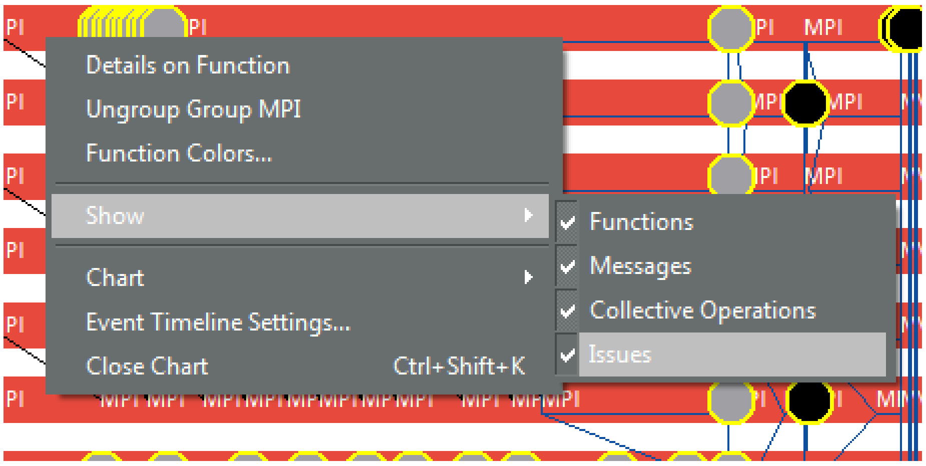
2.4.1. Event Timeline Correctness Checking Reports

On the Event Timeline chart correctness checking reports (CCRs) are displayed as yellow-bordered circles. The color of each circle depends on the type of the particular report: if it is an error then the color is black, if it is a warning - the color is grey.

Event Timeline with CCRs of Both Types:

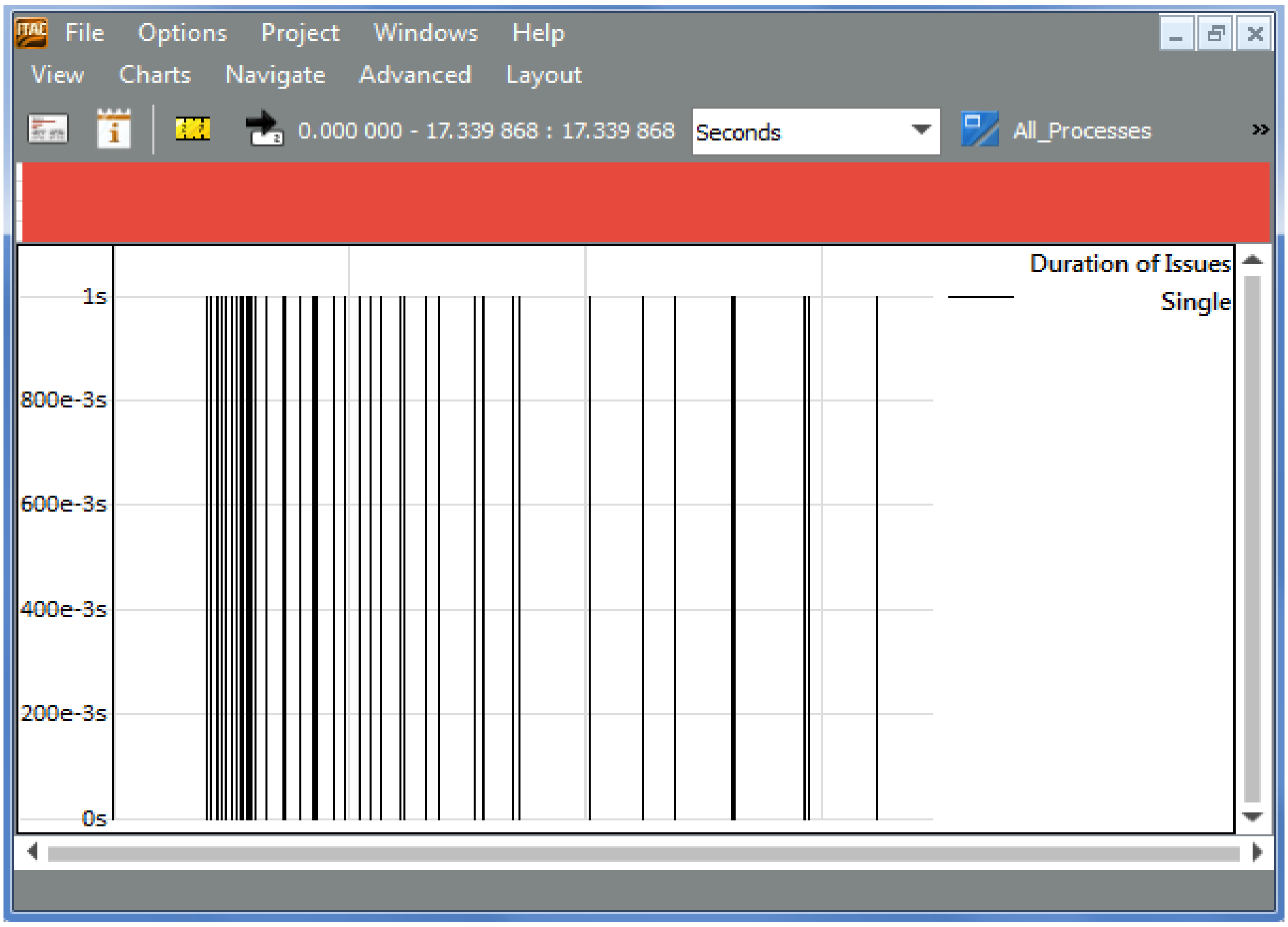


You can turn on/off the display of CCRs through the context menu: right-click the chart, go to the **Show** submenu and check/uncheck the **Issues** item. The display is turned on by default.



2.4.2. Qualitative Timeline Correctness Checking Reports

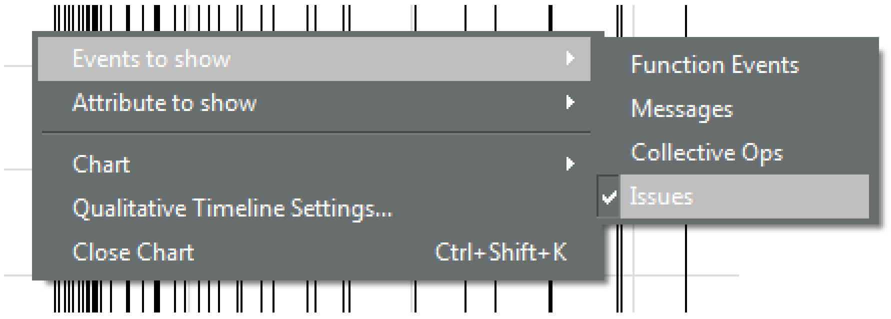
On the Qualitative Timeline chart correctness checking reports (CCRs) are displayed as vertical lines representing the frequency of the CCRs occurrence. The height of each line is constant.



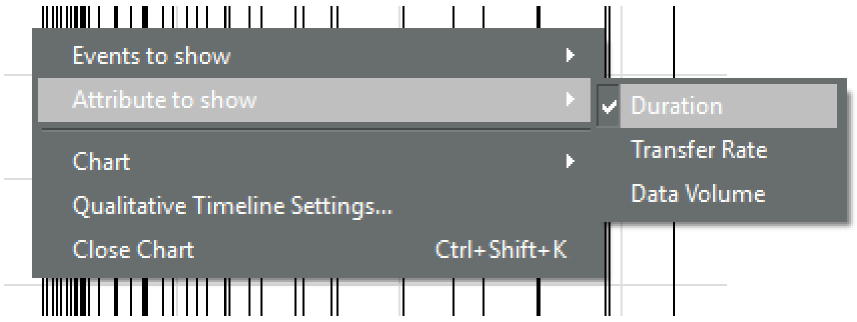
You can turn the display of CCRs on/off through the context menu: for example, right-click the chart, go to the **Events to show** submenu and select **Issues**. The height of each CCR line depends on the level of the report. The lines representing errors are twice higher than the lines representing warnings.

The default event is Duration of Collective Operations.

Submenu "Events to show" in Qualitative Timeline Context Menu:

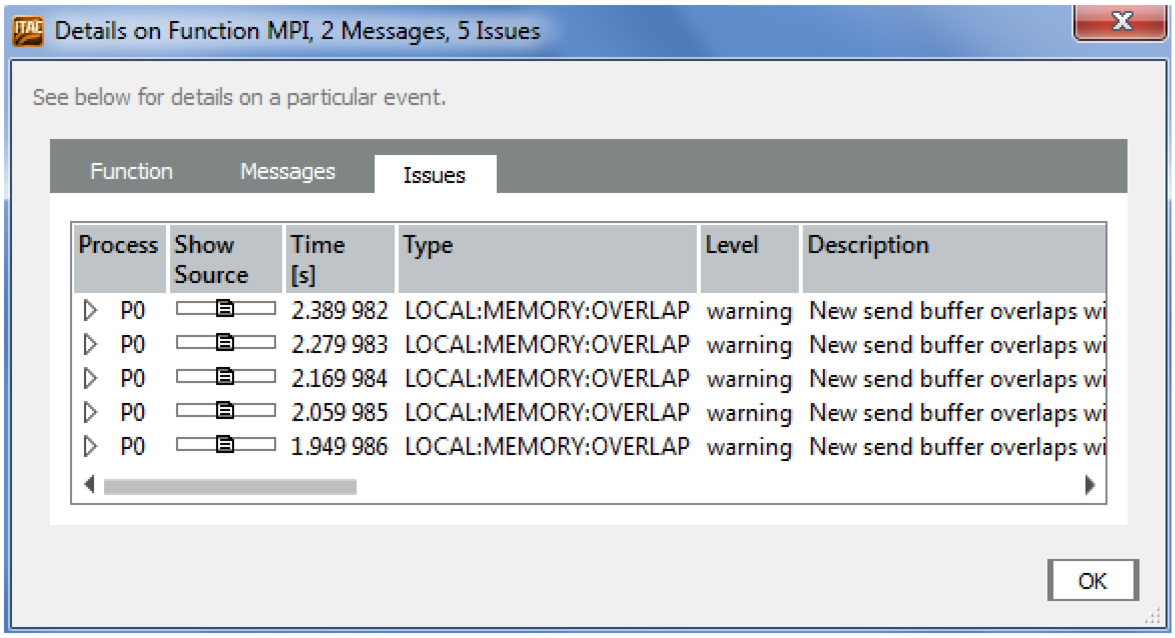


Submenu "Attribute to show" in Qualitative Timeline Context Menu:



2.4.3. Detailed Dialog

If you right-click the point where the correctness checking reposrts (CCRs) are present, you can get information about them in the detailed dialog box. The information contains all fields of the Report Data structure that comes from STF. Each data item may contain five fields that are specifiers for the particular report. You can expand each report item in the Detailed Dialog to get the information from the specifiers.



See below for details on a particular event.

Process	Show Source	Time [s]	Type	Level	Description
▷ P0		2.389 982	LOCAL:MEMORY:OVERLAP	warning	New send buffer overlaps wi
▷ P0		2.279 983	LOCAL:MEMORY:OVERLAP	warning	New send buffer overlaps wi
▷ P0		2.169 984	LOCAL:MEMORY:OVERLAP	warning	New send buffer overlaps wi
▷ P0		2.059 985	LOCAL:MEMORY:OVERLAP	warning	New send buffer overlaps wi
▷ P0		1.949 986	LOCAL:MEMORY:OVERLAP	warning	New send buffer overlaps wi

There can be the following fields:

Field:	**Description:**
Process	Expand the drop-down menu to see processes in which the issue occurred
Show Source	Press this button to get the exact line in the code at which the issue occurred

Time	See the moment of time (in seconds or ticks) at which the issue occurred
Type	See the type of the issue in this string
Level	See the level of the issue in this string. It can be **warning** or **error**.
Entry Time	See the vector containing moments of time for every process involved into the issue
Entry Process	See the vector containing numbers of the processes involved into the issue
Header	See the vector of strings containing descriptions of the issue for a particular process
Call	See the vector of strings containing function calls involved into the issue
Function	See the vector of functions involved into the issue

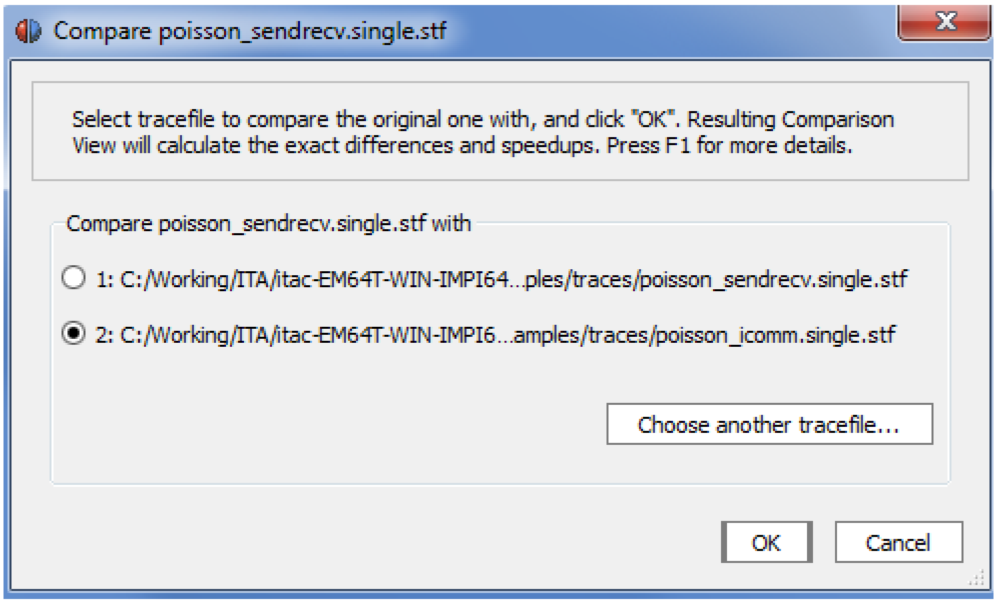
The values of fields from **Entry Time** to **Function** can be different; the sizes of these vectors are equal to each other.

2.5. Comparing Two Trace Files

Comparison View calculates the exact differences and speedups between two runs or between two ranges of the same run.

To open a Comparison View for two trace files, open the files and choose **View > Compare** in one of the trace file views. You can also use the toolbar button.

Select the file to compare the selected file with:



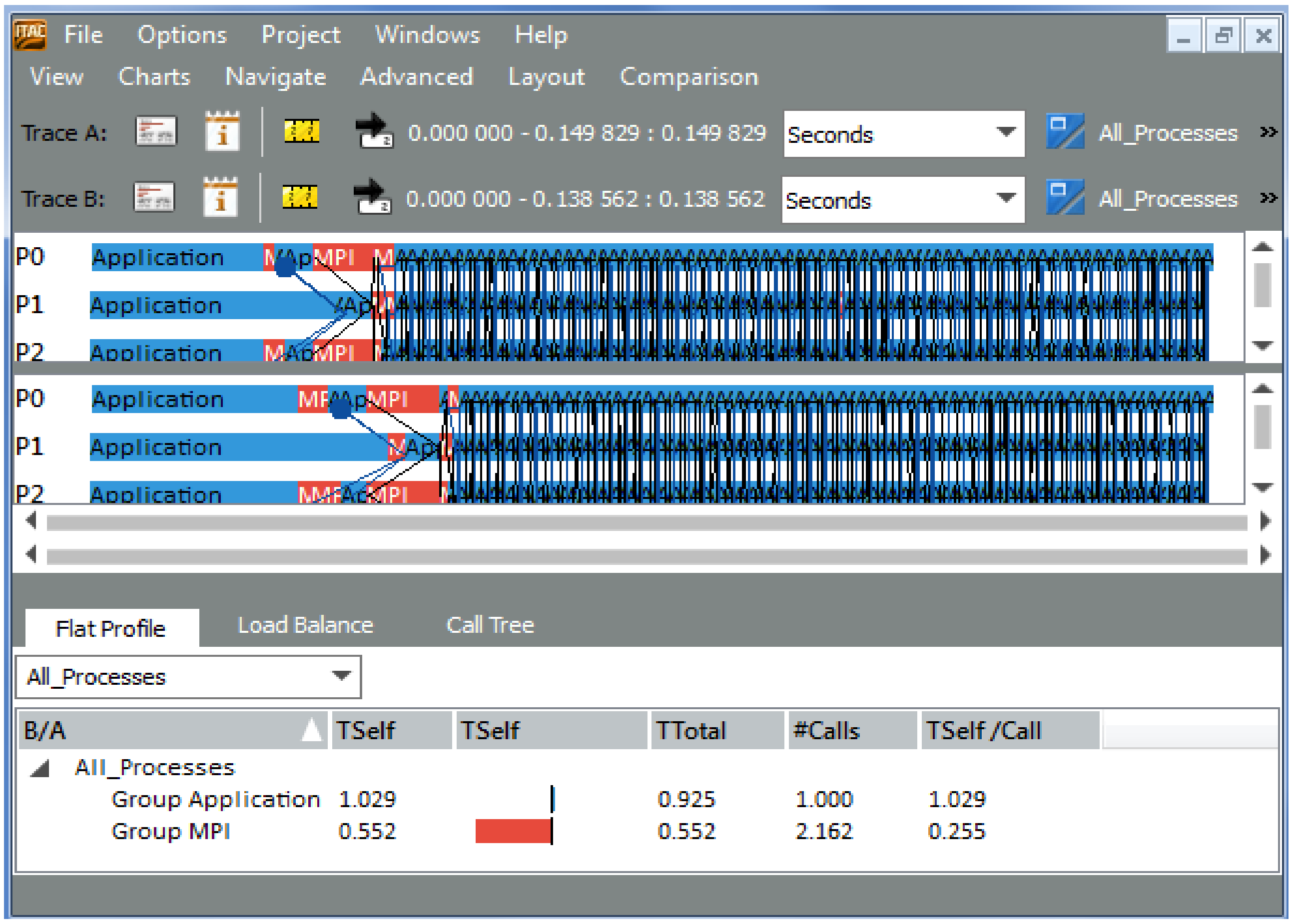
Compare poisson_sendrecv.single.stf

Select tracefile to compare the original one with, and click "OK". Resulting Comparison View will calculate the exact differences and speedups. Press F1 for more details.

Compare poisson_sendrecv.single.stf with

1: C:/Working/ITA/itac-EM64T-WIN-IMPI64...ples/traces/poisson_sendrecv.single.stf [✓ 2: C:/Working/ITA/itac-EM64T-WIN-IMPI6...amples/traces/poisson_icomm.single.stf]

Comparison View appears:



Trace A: 0.000 000 - 0.149 829 : 0.149 829 Seconds All_Processes

Trace B: 0.000 000 - 0.138 562 : 0.138 562 Seconds All_Processes

All_Processes

B/A	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes					
Group Application	1.029		0.925	1.000	1.029
Group MPI	0.552	🟥	0.552	2.162	0.255

Right below the View menu bar, there are two toolbars with labels **Trace A** and **Trace B**. These labels denote two compared trace files.

NOTE

The Comparison View inherits the time interval, aggregation and filter settings from the compared trace files.

By default, **Comparison** dialog box consists of three Charts: Event Timelines for Trace A and Trace B and a Comparison Function Profile.

In the a view of a single trace file, all charts have the same time interval, aggregation and filters. It refers to the Comparison View Charts as well. But since you compare two trace files, the Comparison View holds two sets of time interval, aggregation and filters, one for each compared trace file.

To compare timelines, use Comparison Charts. The **Charts** menu of a Comparison Chart contains comparing variants of the profiles that calculate differences and speedups between the two runs. For explanations, see Comparison Charts.

To control the comparison view, go to **View Menu** > **Comparison** and configure the settings according to your needs.

Set This Option:	**To Do This:**
Same Time Scaling	Set this option to use the same scaling for all timelines. For example, when you look at 2 seconds of file A and 4 seconds of file B the timelines for A are shortened so that they occupy half the width of the timelines for B to provide easier visual comparison. If the time intervals for A and B differ by more than a factor of hundred, this setting is ignored and the timelines are aligned as

	usually to avoid numeric exceptions and distorted diagrams.
Synchronize Navigation Keys	This option controls the behavior of the navigation keys in comparison mode. By default, pressing a navigation key affects only the Chart in focus. For example, if the Chart belongs to file A then the time interval changes for file A only. Set this option to configure the navigation keys to affect both of the zoom stacks(for more information, refer to Zoom Stack).
Synchronize Mouse Zoom	Set this option to configure the zoom function so that it operates on both timelines simultaneously.

See Also

Comparison Charts
Zoom Stack

2.5.1. Mappings in Comparison Views

It is important to compare only comparable things. When comparing two program runs you can try to compare the time that a process A.P0 spent in a function in run A to the time of another process B.P0 in run B with or without caring for the fact that B.P0 did only half of the work because run B used twice as much processes.

It is quite easy to see that depending on the domain decomposition or load balancing that is done in the application the meaningful mapping between the processes of two runs cannot be determined automatically. There might be even no such mapping: imagine comparing a run that did a domain decomposition of a cube into 8 processes 2x2x2 with a run that used a 3x3x3 decomposition.

But functions and function groups can be mapped between the runs just by their fully qualified name. This works as long as the structure of modules, namespaces and classes is not changed dramatically.

It is next to impossible to even enumerate all combinations of parameters that might have changed between two runs. To foresee all these cases in terms of automatically adapting GUI does not look promising.

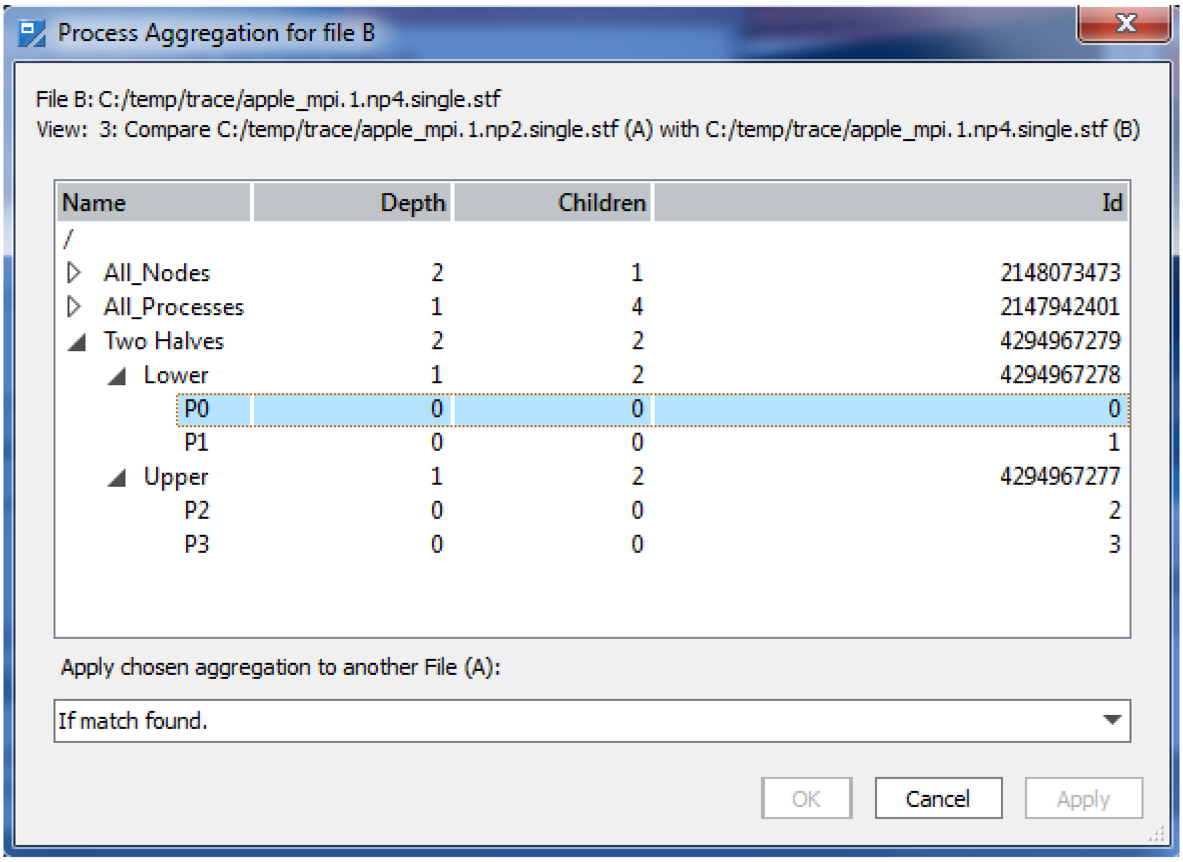
Based on these considerations the mappings of processes, functions, communicators and message tags between the runs are handled differently:

Communicators are mapped by their Ids. Message tags are mapped literally by their value.

The mapping of processes and process groups is controlled by choosing the process aggregations for both files as outlined in Mapping of Processes.

The mapping between functions and function groups is handled automatically as outlined in Mapping of Functions.

See a suitable Process Group for the comparison between a 2 and a 4 processor run in the Process Aggregation dialog box:


Process Aggregation for file B

File B: C:/temp/trace/apple_mpi.1.np4.single.stf
View: 3: Compare C:/temp/trace/apple_mpi.1.np2.single.stf (A) with C:/temp/trace/apple_mpi.1.np4.single.stf (B)

Name	Depth	Children	Id
/			
▷ All_Nodes	2	1	2148073473
▷ All_Processes	1	4	2147942401
◢ Two Halves	2	2	4294967279
◢ Lower	1	2	4294967278
P0	0	0	0
P1	0	0	1
◢ Upper	1	2	4294967277
P2	0	0	2
P3	0	0	3

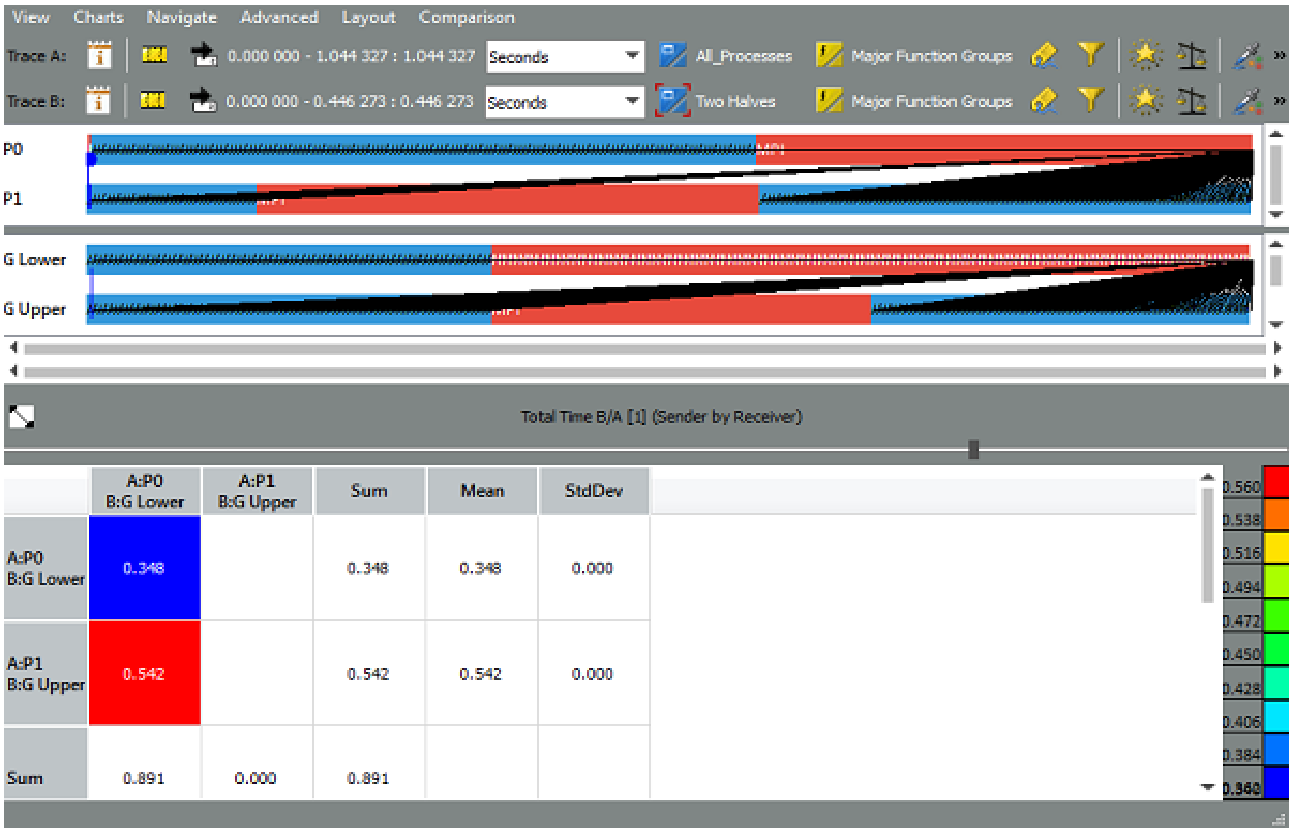
Apply chosen aggregation to another File (A):

If match found.

Mapping of Processes

Assume that run A had A.P0, A.P1 and run B had B.P0, B.P1, B.P2, B.P3 and assume that A.P0 did the same work as B.P0 and B.P1 and A.P1 did the same work as B.P2 and B.P3.

To get a meaningful Comparison Message Profile under these assumptions, choose the aggregation as shown in here and here:



Trace A: 0.000 000 - 1.044 327 : 1.044 327 Seconds All_Processes Major Function Groups

Trace B: 0.000 000 - 0.446 273 : 0.446 273 Seconds Two Halves Major Function Groups

Total Time B/A [1] (Sender by Receiver)

	A:P0 B:G Lower	A:P1 B:G Upper	Sum	Mean	StdDev
A:P0 B:G Lower	0.348		0.348	0.348	0.000
A:P1 B:G Upper	0.542		0.542	0.542	0.000
Sum	0.891	0.000	0.891		

0.560	🟥
0.538	🟧
0.516	🟨
0.494	🟩
0.472	🟩
0.450	🟩
0.428	🟩
0.406	🟦
0.384	🟦
0.362	🟦

This image demonstrates the comparison of Run A (with 2 processes) with Run B (with 4 processes).

The message profile shows the average transfer rate quotient B/A. The Comparison Message Profile (and in fact the whole Comparison View) maps the senders and receivers of the two runs onto each other in the following way: child number i of run A's process aggregation is always mapped to (compared with) child number i of run B's process aggregation.

Mapping of Functions

Functions of the two trace files A and B are mapped onto each other by their fully qualified names. These names contain not only the mere function name, but a hierarchical name constructed by the Intel® Trace Collector with the use of any information about modules, name spaces and classes that are available at trace time.

For example the fully qualified name of MPI_Allreduce is MPI:MPI_Allreduce because the Intel Trace Collector puts all MPI functions into the group MPI. Function groups defined by the user in the Intel Trace Analyzer have no influence on these full function names. The function aggregation shows the fully qualified name of a function in a small tooltip window when the mouse hovers over an entry.

The mapping of function groups is a little more subtle. For function groups that are within the hierarchy of the automatically created function group **Major Function Groups** in file A it is tried to find a matching group in B with the same name and nesting level in the corresponding hierarchy in B.

It works quite well for automatically generated groups. For example MPI is always mapped to MPI even if the groups differ because the two program runs did use a different subset of MPI calls. The same is true for groups that were created by instrumentation using the API provided by the Intel® Trace Collector.

When you create new function groups either by using the Function Aggregation dialog box or the ubiquitous context menu entries to ungroup existing function groups for one file then there will be created matching groups for the other file. You can find these read only groups under the header **Generated Groups** in the Function Aggregation dialog box.

See Also

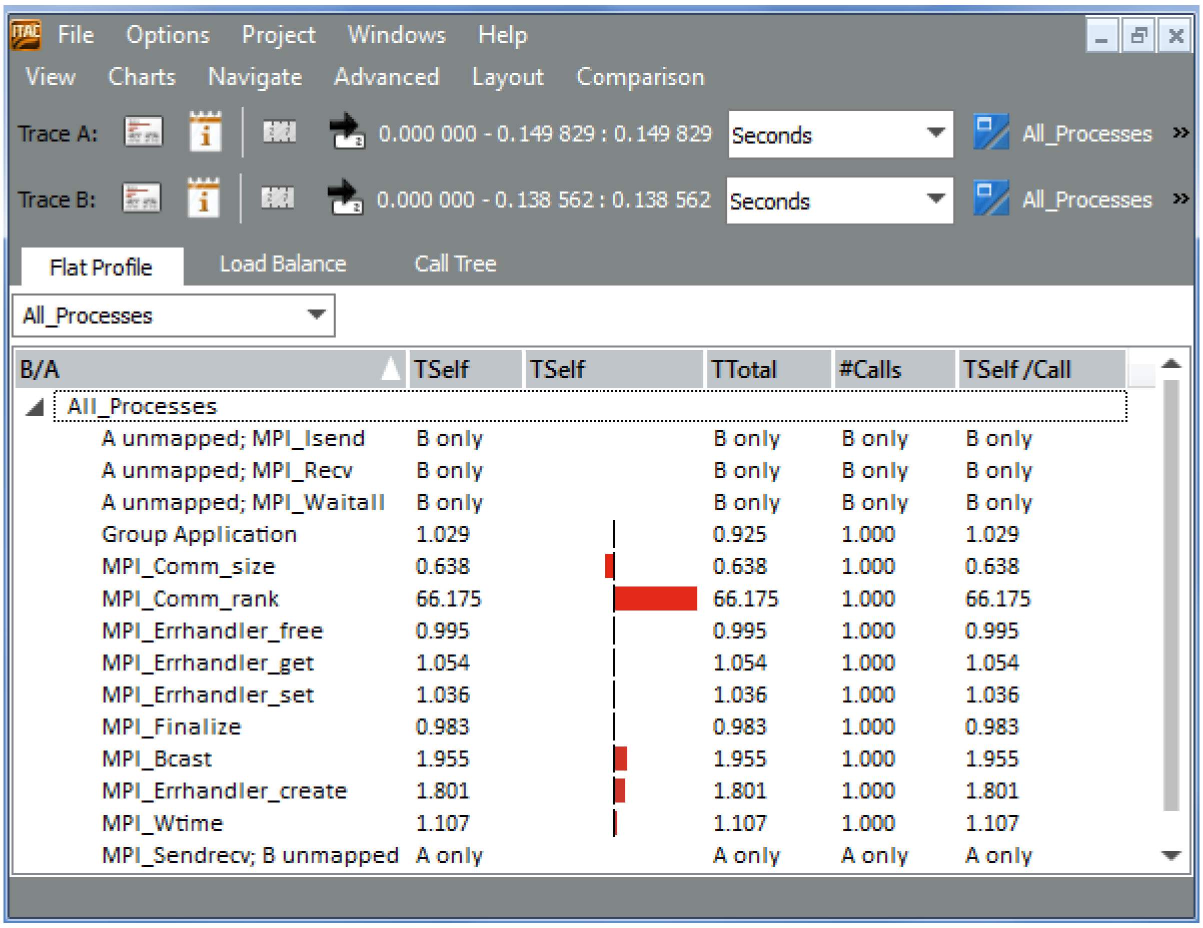
Function Aggregation

2.5.2. Comparison Charts

Comparison Function Profile

The Comparison Function Profile is very similar to the regular Function Profile. But it does not have the pie diagrams in the **Load Balance** tab and there is no **Call Graph** tab at all.

Comparison Function Profile Chart with ungrouped MPI functions:


File Options Project Windows Help
View Charts Navigate Advanced Layout Comparison
Flat Profile
Load Balance
Call Tree

Trace A: 0.000 000 - 0.149 829 : 0.149 829 Seconds All_Processes

Trace B: 0.000 000 - 0.138 562 : 0.138 562 Seconds All_Processes

All_Processes

B/A	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes					
A unmapped; MPI_Isend	B only		B only	B only	B only
A unmapped; MPI_Recv	B only		B only	B only	B only
A unmapped; MPI_Waitall	B only		B only	B only	B only
Group Application	1.029		0.925	1.000	1.029
MPI_Comm_size	0.638		0.638	1.000	0.638
MPI_Comm_rank	66.175		66.175	1.000	66.175
MPI_Errhandler_free	0.995		0.995	1.000	0.995
MPI_Errhandler_get	1.054		1.054	1.000	1.054
MPI_Errhandler_set	1.036		1.036	1.000	1.036
MPI_Finalize	0.983		0.983	1.000	0.983
MPI_Bcast	1.955		1.955	1.000	1.955
MPI_Errhandler_create	1.801		1.801	1.000	1.801
MPI_Wtime	1.107		1.107	1.000	1.107
MPI_Sendrecv; B unmapped	A only		A only	A only	A only

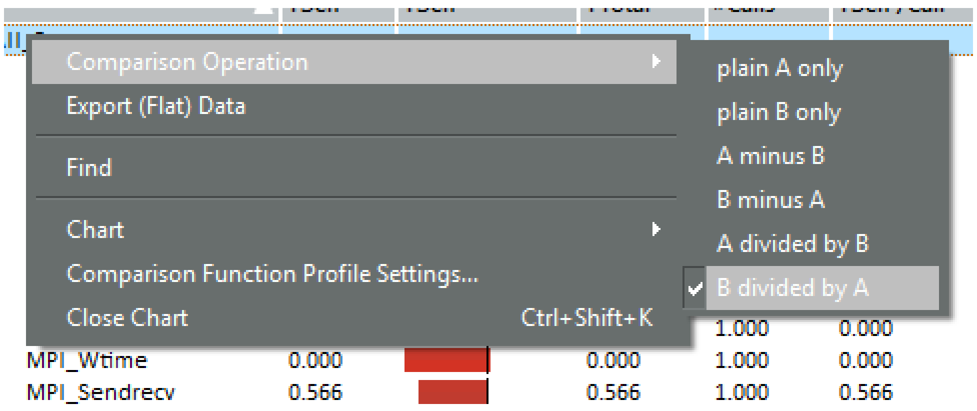
The column headers are the same as in the regular **Function Profile** with the exception of the first column. In the **Comparison Function Profile** this column header displays the currently selected comparison operation. To change a comparison operation, right click on the Chart and choose **Comparison Operation**.

Wherever the regular Function Profile shows the name of a function, function group, process, or process group, the Comparison Function Profile shows either a pair of mapped names in the form of "NameA; NameB" if the names are different, "A unmapped; NameB" or vice versa if there is no mapping for one file, or "Name" if the names are equal for both files.

For example, MPI_Sendrecv in A was replaced by MPI_Isend, MPI_Recv and MPI_Waitall in B. The first column of the profile shows which functions are present in which trace; the other columns indicate which of the traces contain valid data - when there is valid data only for one file, no meaningful comparison can be done.

If you choose **Major Function Groups** and **All Functions** for the function aggregation in both runs, all fields show remarks indicating that comparison is impossible because of lacking data - at least as long as the current comparison operation needs data from A and B. If you switch to a comparison operation of A or B, the respective fields show values even when the function aggregations do not fit.

The Comparison Function Profile context menu has all the entries of the regular Function Profile context menu with the addition of a possibility to choose the comparison operation:



The Comparison Function Profile **Settings** dialog is the same as the one for the regular Function Profile with an additional tab that enables you to switch the comparison operation.

See Also

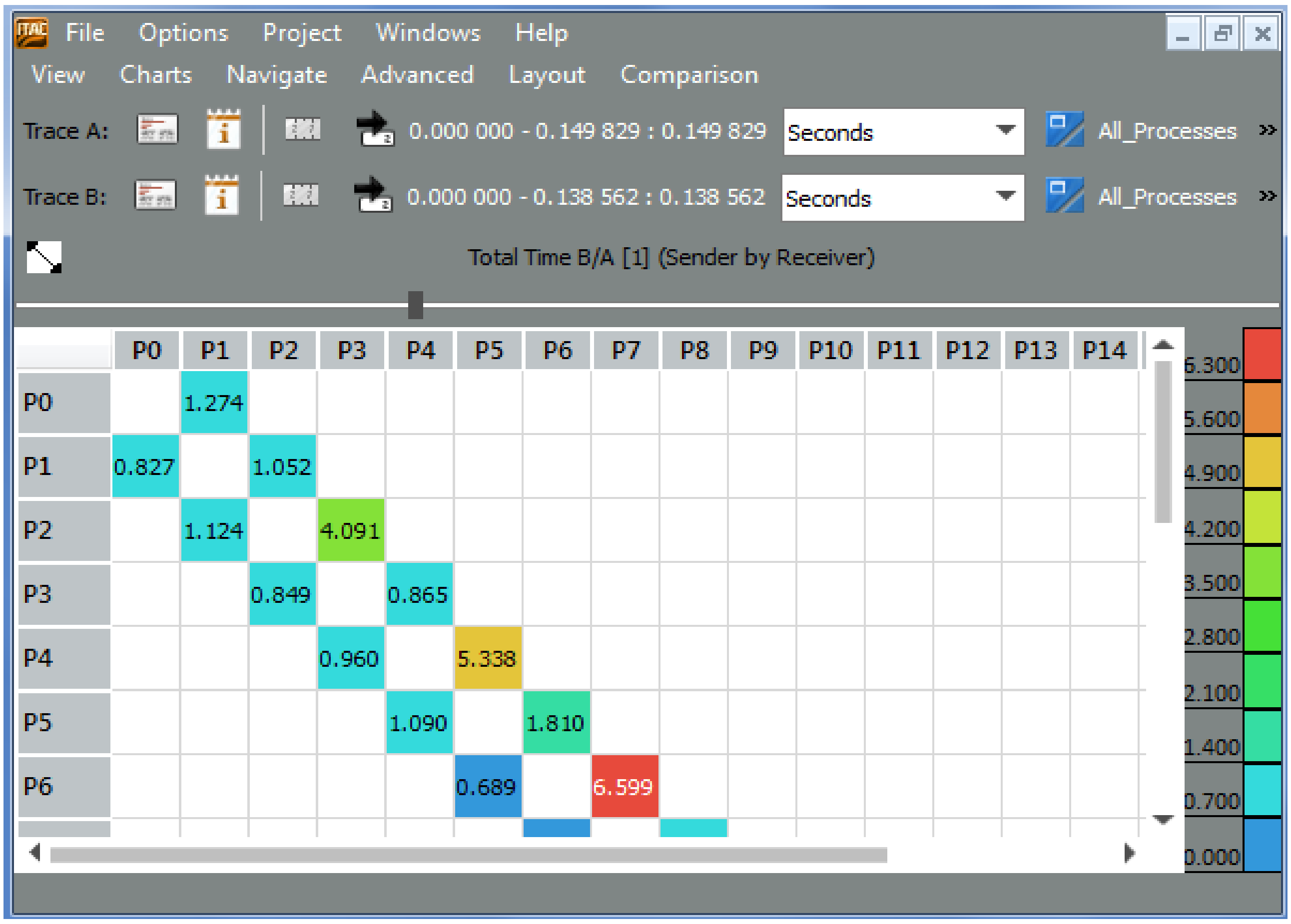
Function Profile Context Menu
Function Profile Settings

Comparison Message Profile

The Comparison Message Profile is very similar to the regular Message Profile. The values shown in the cells are calculated using the currently selected comparison operation shown in the title of the Chart. You can switch the comparison operation from the context menu or from the settings dialog. If the comparison operation makes use of A and B and only one of the values is present for a given cell, this cell is labeled with **A** or **B** to indicate for which trace file the data is present. If there is no data at all, the cell is empty.

Some groupings depend on the mapping rules in which they either use processes, process groups or communicators as row or column labels. If differing labels are mapped onto the same column or row, then the label is made up of two lines with the first line holding the label for file A and the second for file B. This can happen for the groupings **Sender**, **Receiver**, **Sender/Receiver** and **Communicator**.

For example, see the Comparison Message Profile with `poisson_sendrecv.single.stf` as run A and `poisson_icomm.single.stf` as run B:



Trace A: 0.000 000 - 0.149 829 : 0.149 829 Seconds All_Processes

Trace B: 0.000 000 - 0.138 562 : 0.138 562 Seconds All_Processes

Total Time B/A [1] (Sender by Receiver)

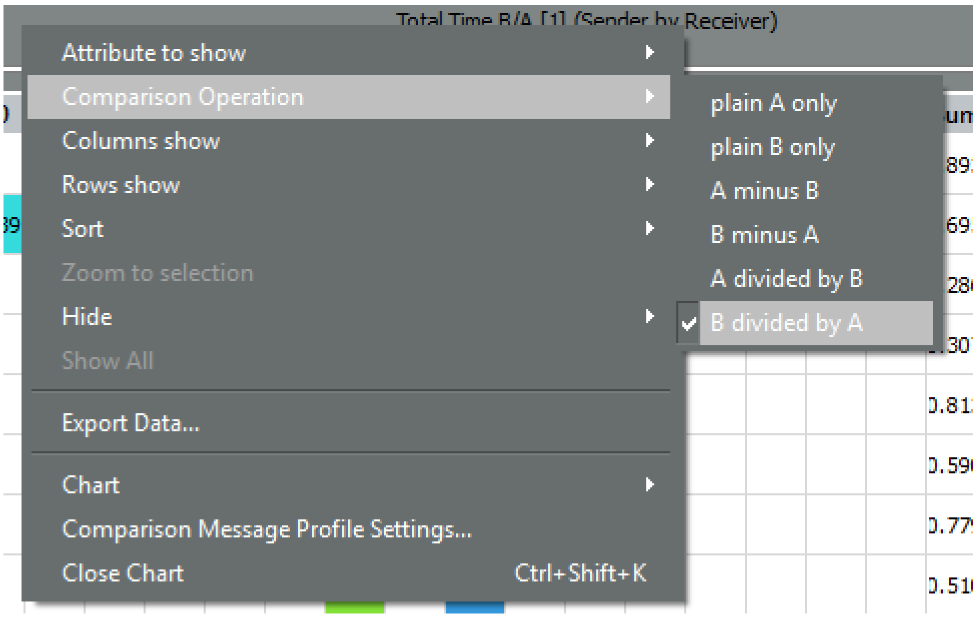
	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
P0		1.274													
P1	0.827		1.052												
P2		1.124		4.091											
P3			0.849		0.865										
P4				0.960		5.338									
P5					1.090		1.810								
P6						0.689		6.599							

6.300	🟥
5.600	🟧
4.900	🟨
4.200	🟨
3.500	🟩
2.800	🟩
2.100	🟩
1.400	🟩
0.700	🟦
0.000	🟦

There can be cases where a whole row or column does not apply to one of the traces. In these cases the double line label holds **A missing** or **B missing**. This can happen for the groupings **Sender**, **Receiver** and **Sender/Receiver** that may have unmappable TGroups in the labels, or for groupings like **Tag**, **Communicator** or **Volume** that directly depend on the data that is present in the trace. You can use the Comparison Message Profile **Settings** dialog box to suppress such rows and columns, but they are shown by default.

The Comparison Message Profile context menu contains all the entries of the regular Message Profile context menu with the addition of the **Comparison Operation** entry, from the submenu of which you can select the type of comparison operation.

The Comparison Message Profile **Settings** dialog box has all the entries of the regular Message Profile Settings. But its **Preferences** tab has two additional check boxes to suppress rows or columns that only apply to one trace, and the **Data** tab has an additional combo box to choose the comparison operation:



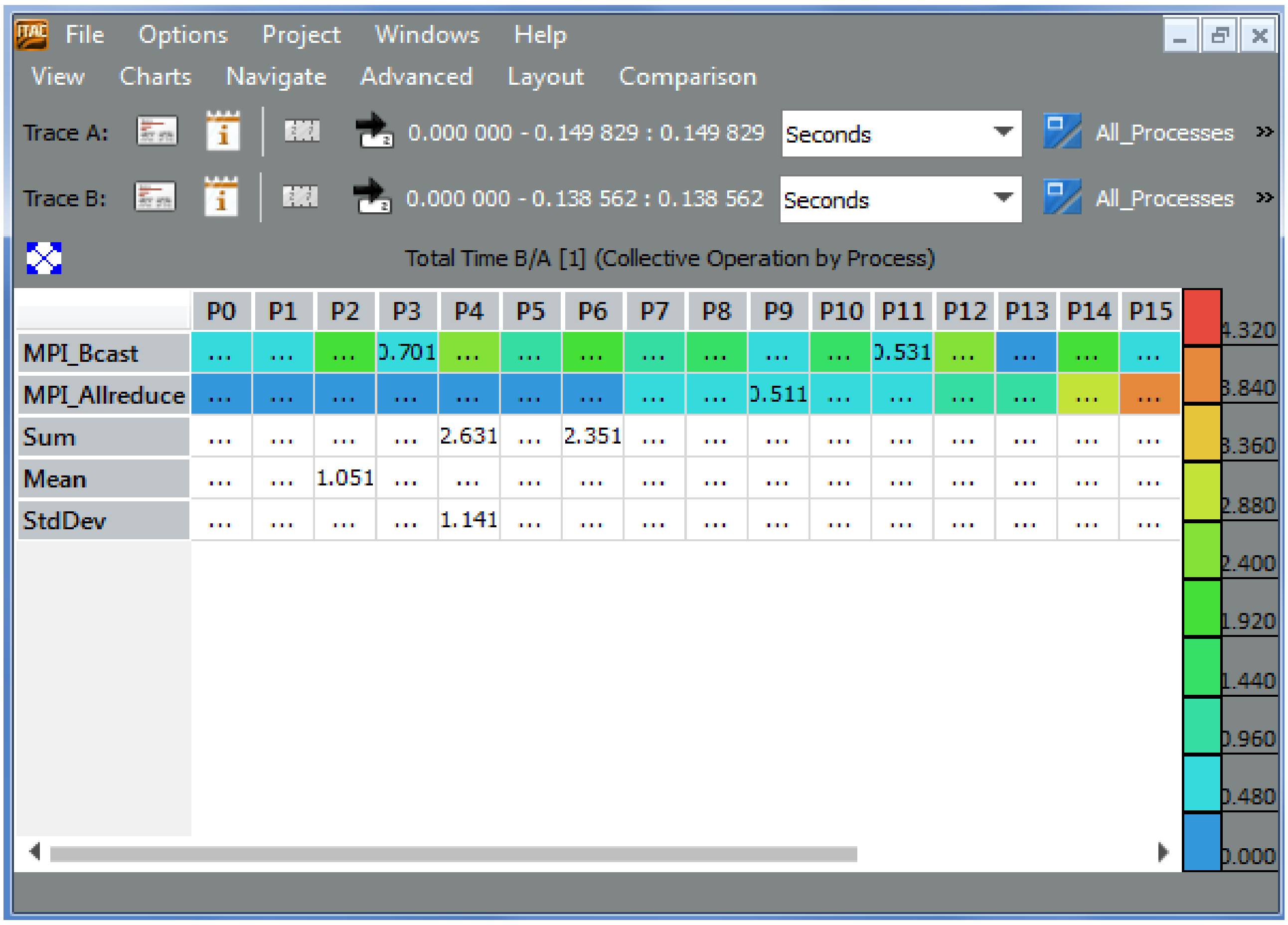
See Also

Message Profile Context Menu
Message Profile Settings

Comparison Collective Operations Profile

The Comparison Collective Operations Profile is similar to the regular Collective Operations Profile. The values shown in the cells are calculated using the currently selected comparison operation shown in the title of the Chart. The comparison operation can be switched from the context menu or from the settings dialog.

Regarding missing values and unmappable column or row labels the Comparison Collective Operations Profile behaves identically to the Comparison Message Profile.



Trace A: 0.000 000 - 0.149 829 : 0.149 829 Seconds All_Processes

Trace B: 0.000 000 - 0.138 562 : 0.138 562 Seconds All_Processes

Total Time B/A [1] (Collective Operation by Process)

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
MPI_Bcast	...	...	...	).701	...	...	...	...	...	...	...	).531	...	...	...	...
MPI_Allreduce	...	...	...	...	...	...	...	...	...	).511	...	...	...	...	...	...
Sum	...	...	...	...	2.631	...	2.351	...	...	...	...	...	...	...	...	...
Mean	...	...	1.051	...	...	...	...	...	...	...	...	...	...	...	...	...
StdDev	...	...	...	...	1.141	...	...	...	...	...	...	...	...	...	...	...

The Comparison Collective Operations Profile context menu has all the entries of the regular Collective Operations Profile Context Menu with the addition of the **Comparison Operation** entry. Use the submenu of this entry to select the type of comparison.

The **Settings** dialog box of the Comparison Collective Operations Profile is similar to the regular Collective Operations Profile Settings. But you can also use the **Preferences** tab of this Settings dialog box to suppress rows or columns that apply to one trace only, and the **Data** tab to choose the comparison operation.

See Also

Comparison Message Profile
Collective Operations Profile Context Menu
Collective Operations Profile Settings

2.6. Interoperability with Intel® VTune™ Amplifier XE and Intel® Advisor XE

To simplify interoperability of Intel® Trace Analyzer with Intel® VTune™ Amplifier XE and Intel® Advisor XE, Intel Trace Analyzer can generate a command line for `mpirun` / `mpiexec.hydra` targeted at analyzing specific processes with Intel VTune Amplifier or Intel Advisor XE. It takes your original command line and modifies it in a way that you can run the mentioned tools with it.

Both Intel VTune Amplifier and Intel Advisor serve to improve the node-level performance of your application. By default, Intel Trace Analyzer prepares command lines for the following types of analyses:

- Intel® VTune™ Amplifier XE: *hotspots analysis* – helps you understand application flow and identify sections of code that get a lot of execution time (hotspots).

- Intel® Advisor XE: *survey analysis* – helps you choose possible places to add vectorization or threading parallelism.

See documentation for the respective tool for more details.

NOTE

This feature requires Intel® MPI Library version 5.0.2 or newer installed on your system, and is currently unavailable on Windows* OS.

Intel Trace Analyzer can generate the command lines for:

- A process identified by Intel Trace Analyzer as the most CPU-bound one. The command lines are generated automatically and displayed on the Summary Page. If the command lines are not available, links to the Intel VTune Amplifier and Intel Advisor documentation is displayed.
- Manually selected processes. To generate the command lines open the **Command line for Intel® VTune™ Amplifier XE/Intel® Advisor XE** dialog from one of the following menus:
    1. **Advanced > Command line for VTune Amplifier...** or **Advanced > Command line for Advisor...**
    2. **Advanced > Process Aggregation**: context menu of the selected process or processes.
    3. **Event Timeline** chart: context menu of the selected process bar or process label. If your application is multithreaded, the command line will contain only process number rather than thread number.
    4. **Function profile > Load Balance** tab: context menu of the selected process.

For example, if your original command line is:

```
$ mpirun –trace –n 128 a.out
```

And you need to analyze processes 33 and 44, the generated command lines will look as shown below.

Intel® VTune™ Amplifier XE:

```
$ mpirun –gtool “amplxe-cl –collect hotspots -r result:33,44” –n 128 a.out
```

Intel® Advisor XE:

```
$ mpirun –gtool “advixe-cl –collect survey:33,44” –n 128 a.out
```

Use the generated command line to analyze the specified processes with Intel VTune Amplifier, or Intel Advisor.

See Also

Command line for Intel® VTune™ Amplifier XE and Intel® Advisor XE Dialog Box
Process Aggregation
Event Timeline
Load Balance

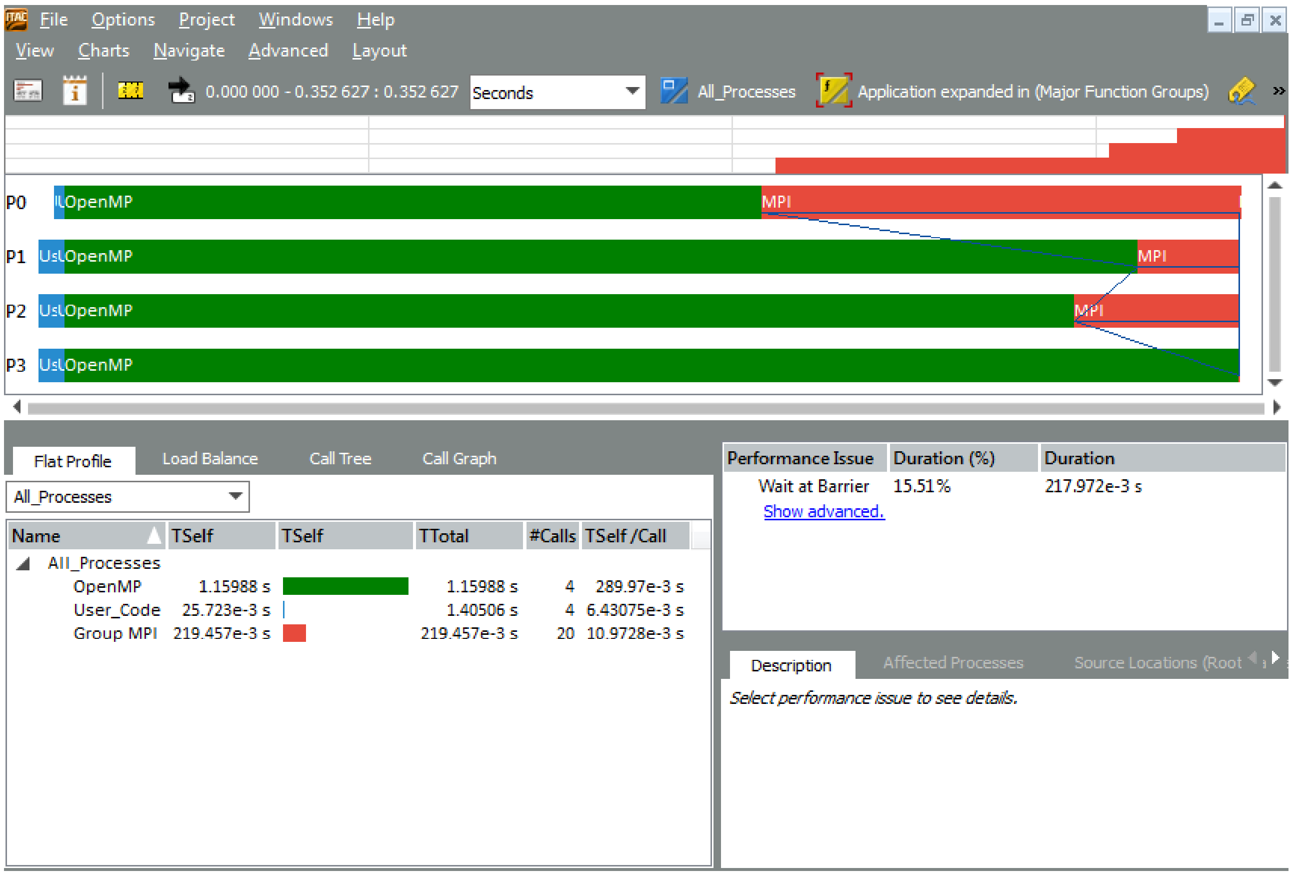
2.7. OpenMP* Regions Support

Intel® Trace Analyzer can display information about OpenMP* parallel regions in your application, if it is contained in the trace file. For details on how to store this information in your trace file, see the *Intel® Trace Collector User and Reference Guide*.

Information about OpenMP regions is available in the following views:

- Summary Page - displays the percentage of OpenMP time in the application time
- Charts: Event Timeline, Function Profile

OpenMP regions are part of the application code. Therefore, to see the OpenMP regions in the charts, you should ungroup the Application group. In one of the charts right-click on **Group Application** and select **Ungroup Application**. OpenMP regions are colored green:



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes					
OpenMP	1.15988 s	🟩	1.15988 s	4	289.97e-3 s
User_Code	25.723e-3 s		1.40506 s	4	6.43075e-3 s
Group MPI	219.457e-3 s	🟥	219.457e-3 s	20	10.9728e-3 s

Performance Issue	Duration (%)	Duration
Wait at Barrier	15.51%	217.972e-3 s

See Also

Summary Page
Event Timeline
Function Profile

2.8. Custom Plug-in Framework

The Custom Plug-in Framework (CPF) enables you to write your own simulators using the provided API. For example, Ideal Interconnect Simulator (IIS) is implemented using this framework. Custom developed simulators can interact with trace files from the Intel® Trace Analyzer and can model any condition you can think of.

CPF implementation works with trace files for applications that pass correctness checking. To use CPF, install Intel® compilers on the systems where you are building your simulator and where you are running CPF.

2.8.1. Usage Instructions

Use the CLI option `--icpf` to process your trace files with a specific simulator library. You can use the following command line:

```
$ traceanalyzer --cli --icpf [options] <tracefile> --simulator <simulator
library>
```

where `<tracefile>` is the name of your trace file, `<simulator library>` is the name of your simulator.

2.8.2. Developing Simulators

Intel® Trace Analyzer and Collector contains examples of the simulator. You can customize the examples of the simulator library to create your own simulator.

The examples are located in: `<trace_analyzer_install_dir>/examples/icpf`

Use the following files in this directory:

- `h_devsim.cpp` - is the simplest template for developing your own simulator.
- `h_doublesim.cpp` - is an example of a simulator library with doubled duration of all MPI events.
- `Makefile` - is a makefile for Linux* gmake.
- `Makefile_win` - is a makefile for Windows* nmake.

Building a Simulator Library

Build a simulator library as follows:

1. Set up your development environment. If you use a non-Intel® compiler, modify the `Makefile` for Linux* OS or `Makefile_win` for Windows* OS in the simulator development directory. To modify a `Makefile`, open the `Makefile` with any text editor and edit the following line to change the compiler:

   ```
   SIMCOMP = icc
   ```

2. To build the examples of simulator libraries, use the following commands:
   - `make devsim` to build the simplest library
   - `make doublesim` to build the library with doubled duration of all MPI events
   - `make` or `make all` to build both libraries

Writing your Simulator

Before beginning simulator development, you need to understand the virtual class that is inherited by a simulator.

The `CustomPluginFramework.h` header file connects the Intel Trace Analyzer source and other tools such as simulators. This file contains the following information:

1. data structures that are passed to the API
2. public virtual functions that are called by CPF
3. private functions that can be called by any simulator help/aid with simulator development
4. private members that provide a simulator with access to certain aspects of a particular trace file

NOTE

If you change the file, you will not be able to plug a simulator into the API.

You can create your simulator by filling virtual functions in `h_devsim.cpp` (or your `.cpp` file, if you renamed it).

Examples

In this example, duration of all non-blocking sending MPI events in trace file becomes zero time.

```
 virtual void ProcessNonBlockingSend( FuncEventInfo * _event_, P2PInfo *
_message_ )
{
   _event_->_endTime = _event_->_startTime;
}
```

In this example, duration of all collective operations is doubled.

```
virtual void ProcessCollOp( CollOpInfo * _collop_ )
{
   //Traverse vector of CollOpInfo
```

```
   for( U4 k = 0; k < _collop_->_threads->count(); k++)
   {
      (*(_collop_->_threads))[k]->_endTime += ( (*(_collop_->_threads))[k]-
>_endTime - (*(_collop_->_threads))[k]->_startTime );
   }
}
```

If you need to add more inline functions for your simulator, add them only to `h_devsim.cpp` or your `.cpp` file. Do not modify the `h_custompluginframework.h` header file.

NOTE

When modifying end time, ensure that the end time you change is never less than the start time.

NOTE

A simulator stores all time values in ticks. To convert ticks to seconds, use the `_clockResolution` coefficient.

NOTE

When you have trouble with trace file, run it through the Intel Trace Analyzer and Collector correctness checker before filling a bug report.

See Also

CPF Simulator API

3. Intel® Trace Analyzer Reference

3.1. Concepts

3.1.1. Level of Detail

Tracing all available events over time can generate billions of events even for a moderate program runtime of a few minutes and a handful of CPUs. The sheer amount of data is a challenge for any analysis tool that has to cope with this data. This is even worse as in most cases the analysis tool cannot make use of the same system resources as the parallel computer on which the trace was generated.

An aspect of this problem arises when generating graphical diagrams of the event data. Obviously, it is next to impossible to graphically display all the data. Firstly, it would take ages to do that. Secondly, it would depend on round-off errors in the scaling and on the order of the data traversal which events would actually make it to the screen without being erased by others. So it is clear that only representatives of the actual events are shown.

A valid choice would be to paint only every 100th or 1000th event and to hope that the resulting diagram gives a valid impression of the data. But this approach has its problems, because the pattern selects the representatives can interfere with the patterns in the underlying data.

Intel® Trace Analyzer uses a Level of Detail concept to solve this problem. The Event Timeline Chart (as the other timelines) calculates a hint for the analysis that describes a time span that can reasonably be painted and selected with the mouse. This hint is called Resolution. The resolution requested by the timeline takes into account the currently available screen space and the length of the current time interval. Hence a higher screen resolution or a wider timeline results in more data being displayed for the same time interval.

Intel® Trace Analyzer then tries to find a near match for the requested resolution. The exact resolution depends on internals, which will not be discussed here.

Intel Trace Analyzer divides the requested time interval into slots of length resolution. After that, representatives for the function events, the messages and the collectives in these slots are chosen in a deterministic way. If a functions spans more than the given resolution it results in a larger slot.

The representatives for function events are chosen as follows: for each slot and each process (or thread group respectively) there is only a single function event representing the function where the thread or group spent most of its time.

The representatives for messages are chosen as follows: for each tuple (sender, receiver, sender slot, receiver slot) only one message is generated that carries averaged attributes. These attributes are averaged over all messages matching the tuple.

The representatives for collective operations are chosen as follows: for each tuple (communicator, first slot) one collective operation is generated. So it can happen that an operation of type `MPI_Gather` is merged with an operation of type `MPI_Bcast` resulting in a merged operation with no particular type at all (mixed).

To prevent misconceptions, emphasis is given to the fact that the merging of events only applies to the timelines and not to the profiles. The profiles always show sums, minima, maxima or averages over the complete set of events. The calculation of these results does obviously not depend on the screen resolution.

3.1.2. Aggregation

Aggregation reduces the amount of data by aggregating events into thread groups and into function groups.

The following aggregation types are discussed in this topic:

- Thread aggregation

- Function aggregation

Thread Aggregation

A striking example for the benefit of thread groups is a parallel code that runs on a cluster of SMP systems. In fact this scenario was the inspiration to introduce this concept. To analyze the behavior of such an application, the data transfer rate is verified to check if the reached rate is plausible with respect to the data rates that are expected (maybe a fraction of the data rates advertised). Of course the effective and expected data transfer rates differ for messages that travel inside an SMP node (intra-node) and between two SMP nodes (inter-node).

In the Intel® Trace Analyzer selecting Aggregation into the predefined **All_Nodes** process group is enough to make the distinction between intra-node and inter-node messages very easy: in the Message Profile the values for the intra-node messages appear on the diagonal of the matrix.

NOTE

Selecting a process group generally results in displaying the information for the group children (with the notable exception of the function profile). That is the reason why you cannot select single, unthreaded processes or single threads for aggregation.

The hierarchy is quite complicated: threads living on the same core (due to Hyper Threading), threads living on different cores in the same CPU, threads living on the same FSB in different CPUs, threads living in the same SMP box on different FSBs, threads living in different boxes connected by a faster interconnect, threads living in SMP boxes connected by a not so fast interconnect and so on. But such hierarchies allow for deeply nested thread groups.

If you select the thread group representing a single node to concentrate on intra-node effects, then the analysis becomes slower than using the thread group **All_Processes** alone. Why does it happen? First of all, Intel Trace Analyzer does not have to do any aggregation for the **All_Processes** thread group because it is flat (assuming no threads are used). The second is, despite the fact that only a single SMP node is chosen, all other threads go through the analysis and are thrown into the artificially created thread group **Other**. Click on **Advanced > Show Process Group 'Other'** to make this group visible. To speed things up, choose a filter that only lets the threads of the selected SMP node pass.

NOTE

Filtering and Aggregation are orthogonal mechanisms in the Intel Trace Analyzer.

Function Aggregation

Aggregation into function groups enables you to decide on what level of detail to look at the threads or thread groups' activity. In many cases it might be enough to see that a code spends some percent of its time in MPI without knowing in which particular function. In some cases optimizing the serial parts of the program might seem more rewarding than optimizing the communication structure.

However, if the fraction spent in MPI exceeds the expectation, then it is interesting to know in which particular MPI call the time was spent. Function grouping allows exactly this shift in perspective by ungrouping the function group MPI.

While the argumentation given in *Thread Aggregation* for having nested thread groups may not be that compelling, the reason for having nested function groups comes quite clear as soon as there occur nested modules, classes and/or name spaces.

Provided that there are adequate function groups, it is also much easier to categorize code by library or by author. In this way, it is possible to concentrate precisely on the code that is considered tunable while code that is controlled by third parties is aggregated into coarse categories.

NOTE

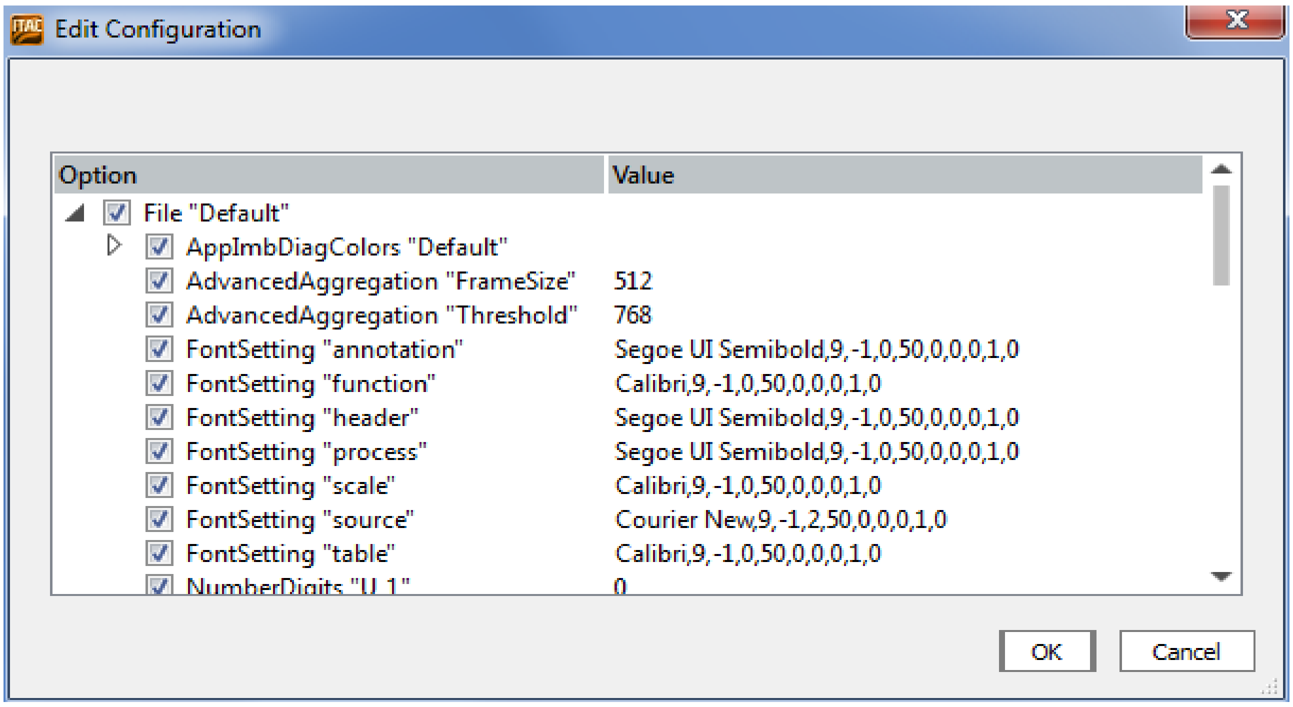
Selecting a function group generally results in displaying the information for the group's children. That is the reason why single functions cannot be selected for aggregation.

In the case of timelines, certain events may not be visible at all times. It does not necessarily mean that they are not there. It can happen because the finer your grouping is, the less time is spent in each individual function/group. On aggregating over processes in a time interval where some processes are idle, nothing is displayed because of the idle state of the processes. Zooming in helps to see these events better. For a better overview, check one of the corresponding profiles. If the timeline and the profile seem to contradict, then the information from the profile is more precise.

3.1.3. Advanced Aggregation

The advanced aggregation function is based on process aggregation and allows lightening the Event Timeline chart for big traces above 1K processes. By default, the function is automatically enabled for traces with above 768 processes, but you can change this threshold using the Edit Configuration Dialog Box.

Advanced Aggregation Settings in the Edit Configuration Dialog:



Option	Value
File "Default"	
AppImbDiagColors "Default"	
AdvancedAggregation "FrameSize"	512
AdvancedAggregation "Threshold"	768
FontSetting "annotation"	Segoe UI Semibold,9,-1,0,50,0,0,0,1,0
FontSetting "function"	Calibri,9,-1,0,50,0,0,0,1,0
FontSetting "header"	Segoe UI Semibold,9,-1,0,50,0,0,0,1,0
FontSetting "process"	Segoe UI Semibold,9,-1,0,50,0,0,0,1,0
FontSetting "scale"	Calibri,9,-1,0,50,0,0,0,1,0
FontSetting "source"	Courier New,9,-1,2,50,0,0,0,1,0
FontSetting "table"	Calibri,9,-1,0,50,0,0,0,1,0
NumberDigits "U 1"	0

When you enable this function, all processes are divided into a set of serial groups. Each group has certain count of processes (except for the last one) and corresponds to a single vertical frame on the **Event Timeline** chart. The default frame size is 512 displayed processes. You can change this value in the Edit Configuration Dialog Box. These processes groups are called **All_Processes_(Visible_*N1*-*N2*)**

1. ***N1*** is the start of displayed range of processes in the group
2. ***N2*** is the end of the range

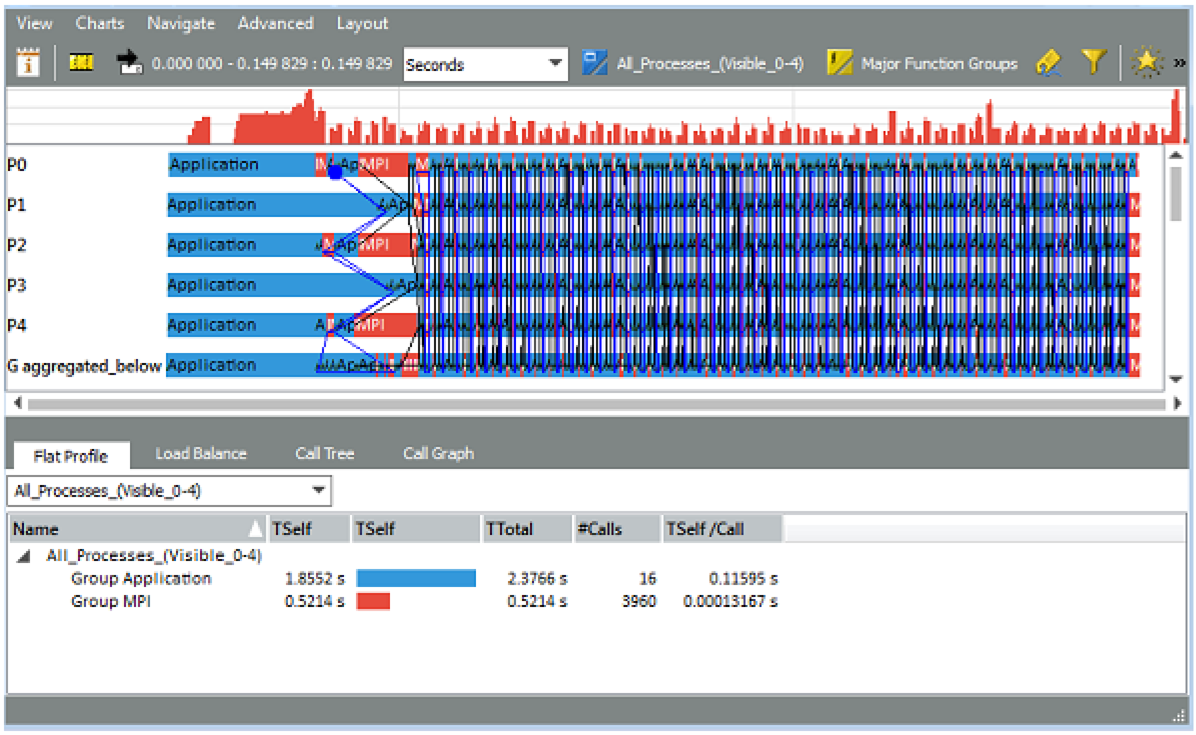
For example, the first advanced aggregation group is called **All_Processes** by default **(Visible_0-4)**.

The advanced aggregation groups contain hidden (non-visible separately) processes as two additionally aggregated groups:

1. aggregated_above
2. aggregated_below

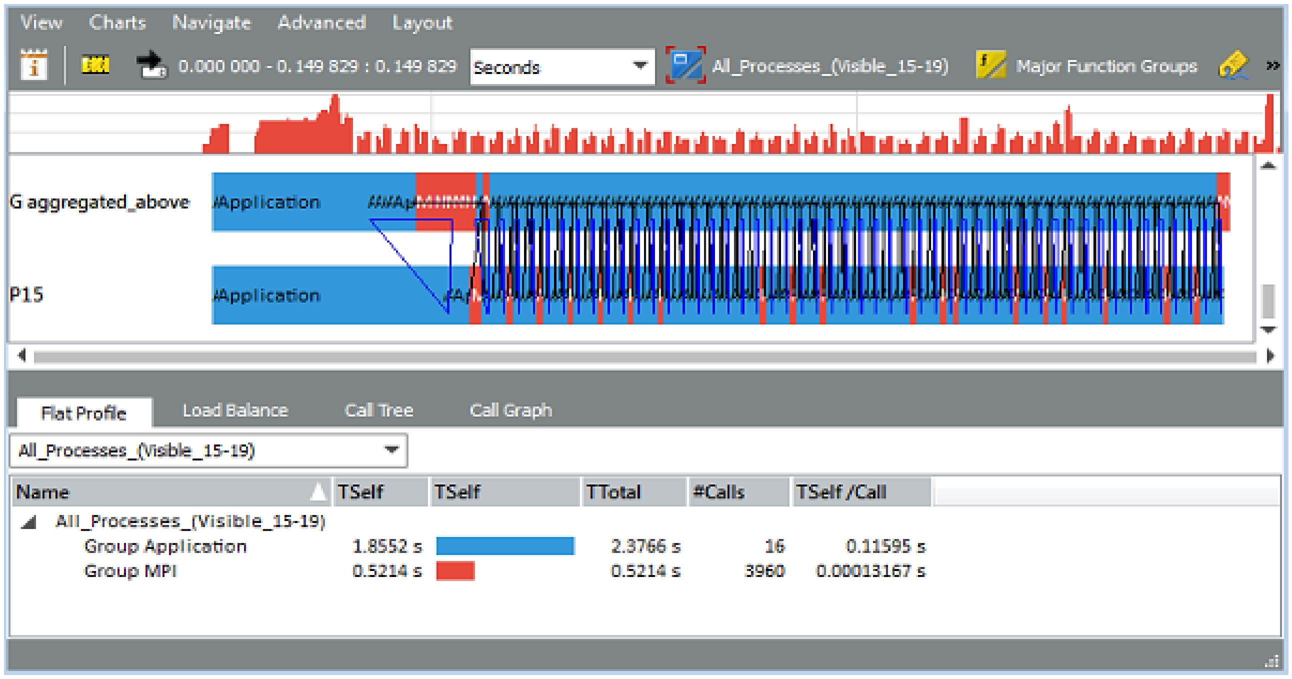
They are displayed on the top and on the bottom of a frame respectively.

Bottom of the First Advanced Aggregation Frame:



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes_(Visible_0-4)					
Group Application	1.8552 s	🟦	2.3766 s	16	0.11595 s
Group MPI	0.5214 s	🟥	0.5214 s	3960	0.00013167 s

Top of the Next Advanced Aggregation Frame:



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
All_Processes_(Visible_15-19)					
Group Application	1.8552 s	🟦	2.3766 s	16	0.11595 s
Group MPI	0.5214 s	🟥	0.5214 s	3960	0.00013167 s

A vertical scrollbar of **Event Timeline** chart enables you to scroll among all advanced aggregation frames.

NOTE

When a jump to the next frame happens, Intel Trace Analyzer loads the corresponding process group. The chart may be not available in a short period.

See Also

Aggregation
Event Timeline
Edit Configuration Dialog Box

3.1.4. Tagging and Filtering

Both concepts use the same grammar to describe their expressions. See Filtering Dialog Box and Tagging Dialog Box for usage hints.

Conceptually there is a different filter for each class of events: function events, messages and collective operations. During analysis the event is recognized when the expression is matched.

The behavior of a filter is determined at the time of its creation. A filter continues filtering until it is changed, even when the thread groups or function groups that it references are changed. Events are treated as belonging to these groups based on the state of the groups at the time the filter was first created.

Tagging

If several events are merged as described in the Level of Detail section, then the merged event is tagged if at least one of the singular events is tagged.

Therefore you can use tagging in particular together with the Qualitative Timeline Chart to find events matching a certain criterion. If a single event out of billions matches the criteria of the tag filter, you see a highlighted peak in the Qualitative Timeline Chart that indicates where to zoom into the trace.

Filtering

If an event is suppressed by filtering then the effect is as if it were never written to the trace file. This is relatively easy to understand for messages and collective operations.

However, if a filter is designed in the way that it suppresses MPI and enables all other functions to pass, then after filtering the Event Timeline looks as if MPI was not called. It looks as if the thread was in the calling function instead of being in MPI. The same is true for the call tree and the call graph of the Function Profile.

What happens if there are functions A, B and C in the trace where A calls B and B calls C and you filter B out? It appears as if A had called C directly. This is quite different from choosing a function aggregation that does not cover B, because that shows the function group other (`other fgroup`) wherever B was shown before. Filtering and Aggregation are independent.

See Also

Qualitative Timeline

3.2. Command Line Interface (CLI)

The Command Line Interface (CLI) to the Intel® Trace Analyzer enables you to process trace files without a GUI.

Use the CLI to:

- Compute profiling data automatically
- Generate pre-computed trace caches for trace files

To enable the CLI, use `--cli` as the first argument to switch off the graphical user interface followed by a trace file name and any other CLI options.

For example, to perform message profile analysis on `trace.stf`, apply filter by zero sender rank and print the output in `messages.log`, enter:

```
$ traceanalyzer --cli --messageprofile --filter=p2pfilter(sender(0)) -o
messages.txt trace.stf
```

If you do not specify the output file, results will be printed in standard output.

To create the cache for `trace.stf` with default resolution, enter:

```
$ traceanalyzer --cli trace.stf -c0 -w
```

A batch file to pre-compute caches might look like this:

```
$ traceanalyzer --cli poisson_icomm.single.stf -c0 -w
$ traceanalyzer --cli poisson_sendrecv.single.stf -c0 -w
$ traceanalyzer --cli vtcounterscopec.single.stf -c0 -w
```

NOTE

The CLI is for expert use and can be changed with any version without notice.

The command line interface provides the following options:

Option Name	Action
`--messageprofile`	Perform message profile analysis.
`--collopprofile`	Perform collective operation profile analysis.
`--functionprofile`	Perform function profile analysis.
`--starttime=TICKS or -sTICKS`	Starting time of the analysis.
`--endtime=TICKS or -eTICKS`	Ending time of the analysis.
`--tgroup=ID or -tID`	Use this thread aggregation.
`--fgroup=ID or -fID`	Use this function aggregation.
`--dump=FILE or -oFILE`	The file where to store the analysis results. If not specified, results are printed in standard output.
`--funcformat`	A string that contains format switchers specifying how the information about functions are printed; the default value is `TFNEIS`. Possible format options: 1. `f` or F - prints the name of the function group 2. `t` or T - prints the name of the thread/process group 3. g or G - prints the number of processes/threads in the group 4. E - prints self time in ticks 5. e - prints self time in seconds 6. `I` - prints total time in ticks 7. `i` - prints total time in seconds 8. n or N - prints the number of calls 9. s or S - prints the source code location (if possible)
`--messageformat`	A string that contains format switchers specifying how the information about point-to-point messages is printed; the default value is `12DdIiXxAauUn`. Possible format options: 1 - prints if the first member of the message is sender and/or receiver 2 - prints if the second member of the message is sender and/or receiver D - prints the summary duration in ticks

	d - prints the summary duration in seconds v or V - prints the summary amount of bytes sent k or K - prints the minimum amount of bytes sent l or L - prints the maximum amount of bytes sent U - prints the minimum duration in ticks u - prints the minimum duration in seconds X - prints the maximum duration in ticks x - prints the maximum duration in seconds I - prints the minimum rate in Bytes/tick i - prints the minimum rate in Bytes/second A - prints the maximum rate in Bytes/tick a - prints the maximum rate in Bytes/second n or N - prints the number of messages
`--collopformat`	A string that contains format options specifying how the information about collective operations is printed; the default value is `12DdIiXxAauUnvwyzlk`. Possible format options: 1 - prints the name of the process group 2 - prints the name of the operation D - prints the summary duration in ticks d - prints the summary duration in seconds U - prints the minimum duration in ticks u - prints the minimum duration in seconds X - prints the maximum duration in ticks x - prints the maximum duration in seconds I - prints the minimum rate in Bytes/tick i - prints the minimum rate in Bytes/second A - prints the maximum rate in Bytes/tick a - prints the maximum rate in Bytes/second v or V - prints the summary amount of bytes sent k or K - prints the minimum amount of bytes sent l or L - prints the maximum amount of bytes sent w or W - prints the summary amount of bytes received y or Y - prints the minimum amount of bytes received z or Z - prints the maximum amount of bytes received n or N - prints the number of collective operations
`--readstats or -S`	Request statistics, if available, instead of trace data.
`--readcache[=FILE] or -r[FILE]`	Read the trace cache from the specified (if provided) or default file.
`--writecache[=FILE] or`	If a trace cache has been built, write it to the specified (if provided) or

`-w[FILE]`	default file.
`--buildcache=RESOLUTION` or `-cRESOLUTION`	Build a trace cache with the specified resolution. The resolution is given in clock ticks. Higher values result in smaller (coarser) cache files, 0 (zero) is used as the default resolution.
`--filter=EXPRESSION` or `-FEXPRESSION`	The filter to use for the analysis, specified as a filter grammar string. `EXPRESSION` may be: `funcfilter`, `p2pfilter`, `collfilter` or their combinations. For details, see the Filter Expression Grammar section.
`--messagefirst=GROUPING`	The first grouping in the message profile analysis result (first dimension of matrix).
`--messagesecond=GROUPING`	The second grouping in the message profile analysis result (second dimension of matrix).
`--collopfirst=GROUPING`	The first grouping in the collective operation profile analysis result (first dimension of matrix).
`--collopsecond=GROUPING`	The second grouping in the collective operation profile analysis result (second dimension of matrix).
`--summary`	Generate the application summary sheet with the format that is described below.
`--icpf [options]` *`<tracefile>`* `--simulator` *`<simulator library>`*	Process a trace file using the specified simulator at runtime. Use the traceanalyzer `-icpf` option to process your trace files using specific simulator library. In `--icpf [options] <tracefile> --simulator <simulator libraray>`, the [options] can be: `-s` <*NUM*> - processes the trace starting at the time (NUM measured in ticks). `-e` <*NUM*> - processes the trace to the end time (NUM measured in ticks). `-w` <*NUM*> - processes the trace based on NUM, 0 for STF, 1 for ASCII, else devnull. `-o` <*new_name*> - trace output file name. `-u` - single file mode. The output file is a single STF. `-h` - prints this message and exits.
`--ideal [options]` *`<tracefile>`*	Produce an ideal trace. Use the traceanalyzer `--ideal` option to idealize a trace by Ideal Interconnect Simulator. In `--ideal [options] <tracefile>`, the `[options]` can be: -s <*NUM*> - processes the trace starting at the time (`NUM` measured in ticks; the default value is 0). `-e` <*NUM*> - processes the trace to the end time (NUM measured in ticks; the default value is the end time of the trace). `-w` <*NUM*> - processes the trace based on NUM, 0 for STF, 1 for ASCII, else devnull (the default value is 0). `-o` <*new_name*> - trace output file name. `-u` - single file mode. The output file is a single STF.

	`-sp` - shows percent progress indicator. `-q` - quiet mode; turns off all output. `-h` - prints this message and exits.
`--breakdowns` *`<real_trace_name>`* *`<ideal_trace_name>`*	Create intermediate *.bdi files that contain all needed information for the Imbalance Diagram.
`--merge` *`<unmerged_trace_name>`* `[`*`<merged_trace_name>`*`]` `[-single] [-delete-raw-data] [-sumdata]`	Merge the raw trace. *`<merged_trace_name>`* - if set this option, then the output trace will have this name; otherwise suffix `.merged` will be added to the original name. `-single` - create a single STF file instead of multiple ones `-delete-raw-data` - delete the raw trace after merging `-sumdata` - create summary data files while merging
`--sumdata` *`<trace_name>`*	Create summary data files from an ordinary trace
`--assist [options]` *`<tracefile>`*	Use the `--assist` option to discover performance problems in your application. To learn more about the Performance Assistant, refer to the Performance Assistant section. In `--assist [options]` *`<tracefile>`* `[options]` can be: `-s` *`<NUM>`* - processes the trace starting at the time NUM measured in ticks; the default value is 0. `-e` *`<NUM>`* - processes the trace to the end time NUM measured in ticks; the default value is the end time of the trace `-h` - prints this message and exits
`--interval=PERCENT or -iPERCENT`	Select the time interval in the trace file to be analyzed. PERCENT represents the percent of time taken from the middle of the trace file. This value may range from 0 to 100 (default). For example, if you set the interval to 20%, and your application time is 10 seconds, only the interval from 4 to 6 seconds will be analyzed.

The application summary sheet consists of a three-line header:

```
<# processes>:<# processes per node>
<application time>:<MPI time>:<IIS time>
<first message size of middle bucket (2)>: \
<first message size of highest bucket (3)>
```

The header is followed by these sets of lines, for each of the top ten functions, sorted by descending total time:

```
<Name of MPI_group>:<# involved processes>
<total time in above func for bucket 1>:<for bucket 2>:<for bucket 3>
<total IIS time in above func for bucket 1>:<for bucket 2>:<for bucket 3>
<count in above func for bucket 1>:<for bucket 2>:<for bucket 3>
<total # bytes in above func for bucket 1>:<for bucket 2>:<for bucket 3>
```

In the application summary sheet, IIS stands for Ideal Interconnect Simulator, which predicts MPI behavior on an ideal interconnect.

You can import the application summary sheet to spreadsheet applications such as Microsoft* Office Excel*. Fields are separated by colons. Unknown values are indicated by **N/A**.

3.3. Custom Plug-in Framework API

Custom Plug-in Framework (CPF) interacts with the simulators through the custom plug-in framework API. The API provides pre-defined functions that the simulators have to implement.

3.3.1. Framework Variables

Variable	Explanation
U4	unsigned 4-byte integer
U8	unsigned 8-byte integer
I4	signed 4-byte integer
I8	signed 8-byte integer
F4	4-byte floating point
F8	8-byte floating point

3.3.2. Data Structures

FuncEventData

The FuncEventData structure represents any function events that occur in a trace on a specific thread and do not span multiple threads. CPF passes a FuncEventData structure to a simulator whenever it encounters any application event, MPI single rank event, or a non-tracing event (such as idle time on a process). The FuncEventData structure also represents a non-blocking sending because a non-blocking sending occurs only on a specific thread.

The only member of FuncEventData that you can modify using a simulator is the end time of the function event. The other parameters are all constant because modifying any of these values would greatly change the fundamental parts of the trace. The constant parameters can be the thread id, function type, size of message (if it is a non-blocking sending), or start time.

NOTE

The end time of this event should never be less than the start time. CPF checks this rule when returned from the simulator.

Variable	Explanation
`_threadID`	The thread id where this function event occurred.
`_func`	The function id of this event. You can look up the id using mapping provided. See Private Functions for details.
`_sizeSent`	The size of the message sent by the thread. This variable is available when the function event is used as part of the collective operation.

`_sizeRecv`	The size of the message recieved by the thread. This variable is available when the function event is used as part of the collective operation.
`_startTime`	The start time of this function event.
`_endTime`	The end time of this function event. You can modify this variable, but make sure that it is not less then `_startTime`.

P2PInfo

The P2PInfo data structure represents a point-to-point (P2P) message that occurs in a trace. It is also passed to non-blocking API function to give the simulator access to different aspects of the point-to-point message. The values are slightly different depending on the type of a point-to-point message that this data structure represents. See Private Functions for details.

The following table lists the P2Pinfo variables. All variables, except the end times of the send and receive messages, are constant because modifying any of these values would drastically change the fundamental parts of the trace.

Variable	**Explanation**
`_sendStartTime`	This variable contains the send start time or the time when the message is sent.
`_sendEndTime`	This variable is only valid on sendrecv non-blocking sends (only in `processnonblockingsend()`) or on sync sends. This variable can be modified, but should never be less then `_sendStartTime`.
`_recvStartTime`	This variable is only valid on sendrecv, sync sends, waitall/waitsomes.
`_recvEndTime`	This variable is valid on all cases, can be modified, but should never be less then `_recvStartTime`.
`_iRecvStartTime`	This variable is only valid when P2P message involves a waitall/waitsome. This variable represents the start time of i-receive that occurs before the waitall/waitsome for this thread.
`_backwards`	This variable indicates that the message send start time > receive end time.
`_sender`	The thread id of the sending thread.
`_receiver`	The thread id of the receiving thread.
`_communicator`	The MPI communicator. The P2P message communication is over; there is a communicator to text mapping provided for the simulator in the API. See Private Functions for details.

`_tag`	The MPI tag associated with the message.
`_messageSize`	The size of the message sent.
`_sendFID`	The identifier of the sending function. You can get more information on the function using the mapping provided. See Private Functions for details.
`_recvFID`	The identifier of the receiving function. You can get more information on the function using mapping provided. See Private Functions for details

CollOpInfo

The `CollOpInfo` data structure represents a collective operation that occurs in the trace. In a trace, there is a collective operation that connects the same event on each thread in the trace to one another. This `CollOpInfo` data structure provides this grouping to the simulator.

All members of the `CollOpInfo` data structure are constant and cannot be modified except for the `FuncEventData` events that represent each thread that is involved in the collective operation. To modify end times of the collective operation on each thread, modify the `FuncEventData` for each thread in the vector in this collective operation. The same rules apply `CollOpInfo` for `FuncEventData`.

Variable	**Explanation**
`_firstTime`	The earliest time that a thread enters a collective operation.
`_communicator`	Communicator on which collective operation is executed.
`_collOpId`	The function id of the collective operation. You can look up what it is using mapping provided. See Private Functions for details.
`_root`	The thread id of the root, valid only if the collective operation is a One-To-All or All-To-One.
`_rootIndex`	The root's index into the thread vector.
`_threads`	The thread vector that represents the individual function events per thread that make up this collective operation.

3.3.3. Public Functions

The following topics list the public API functions and general public functions that CPF calls. When CPF is interacting with the API, it first creates a data structure of type described in Data Structures; then passes a pointer to the simulator's API function. Any modification to this data structure in the API function automatically reflects changes in CPF. These functions do not return any values.

void setup()

```
void setup (const Vector< U4 >, const Vector< U4 >, const Vector< U4 >, const
HashMap< U4, const char* >, const char*, const U8, const U8, const U4)
```

Use this function to pass important information/characteristics about the trace from CPF to the simulator. This function takes care of initialization of the private member variables in Private Members. Do not modify this function.

Function	Description
`virtual void Initialize ( ) = 0;`	Your simulator needs to implement this API function. The function is called before the start of processing a trace file, not to be used as a constructor. This function enables simulator extensions to pre-trace file operations before the executable starts processing the trace file.
`virtual void Finalize ( ) = 0;`	Your simulator needs to implement this API function. The function is called at the very end of processing a trace file, not to be used as a destructor. This function allows simulator extensions to have a way of doing post-trace file operations before the executable exits.
`virtual void ProcessFunctionEvent ( FuncEventInfo * _event_ ) = 0;`	Your simulator needs to implement this API function. This function processes application events that occur in the trace file. The event object is a regular application event in the trace that is processed from the queues.
`virtual void ProcessNonTracingEvent ( FuncEventInfo * _event_ ) = 0;`	Your simulator needs to implement this API function. This function processes non-tracing events that occur in the trace file. This event object is a non tracing event in the trace that is to be processed from the queues. A non-tracing event is an empty white space at beginning of trace before each thread, or parts of a trace file which have tracing turned off.
`virtual void ProcessMPIFunctionEvent ( FuncEventInfo * _event_ ) = 0;`	Your simulator needs to implement this API function. This function processes a single rank MPI event in trace file. This event object is a single rank MPI event in the trace that is to be processed from the queues. A single rank MPI event is an MPI event that only involves one thread.
`virtual void ProcessCollOp ( CollOpInfo * _collop_ ) = 0;`	Your simulator needs to implement this API function. This function processes a collective operation MPI event in trace file. This event object is a collective operation MPI event in the trace that is ready to be processed from the queues. This one object represents all threads involved in collective operation. The simulator does not get separate events for each thread involved in the collective operation.
`virtual void ProcessP2P ( P2PInfo * _message_ ) = 0;`	Your simulator needs to implement this API function. This function processes a point-to-point MPI event in trace file. This event object is a P2P MPI event in the trace that is to be processed from the queues. This function is called for

	sendrecv, synchronous, or normal send to receive MPI operations.
`virtual void ProcessNonBlockingSend ( FuncEventInfo * _event_, P2PInfo * _message_ ) =0;`	Your simulator needs to implement this API function. This function processes a non-blocking send MPI event in a trace file. This function gets an event object that represents the send operation and the corresponding P2P message that goes with that send. This function is only called for non-synchronous sends, or sends that do not rely on the receive thread. The start time of the receive thread may not be known.
`virtual void ProcessP2PWaitAllSome ( Vector<P2PInfo*, ObjectPointerAllocator<P2PInfo*> > * _messages_ ) = 0;`	Your simulator needs to implement this API function. This function processes a waitall and waitsome MPI event in trace file. This function gets a vector of all P2P messages involved with the waitall/waitsome. There is no individual function events corresponding to the send or receive thread involved in this P2P operation.

3.3.4. Private Members

Member	**Explanation**
`Vector< char* > _commandLineParams;`	This vector represents the command-line parameters passed to the simulator by CPF. It is a vector of char pointers of parameters that do not match any keywords for CPF. These parameters may contain invalid values, and need to be checked by simulator.
`char * _stfPath;`	Pathway to the directory of the original STF file loaded by CPF at runtime.
`U8 _stfStart;`	Start time of original STF file.
`U8 _stfEnd;`	End time of original STF file
`U4 _numThreads;`	Number of threads in trace
`F8 _clockResolution;`	Resolution length of clock tick in seconds.
`Vector< U4 > _singleRankFuncs;`	Vector of single rank function id's. This vector is initialized in an `init()` call with the id values of every rank function in the trace file. Ranks that are not in the trace file are initialized with the value -1.
`Vector< U4 > _collOpFuncs;`	Vector of collective operation id's. This vector is initialized in an `init()` call with the id values of every collective function in the trace file. Given a function id number, the corresponding collective operation id is returned.
`Vector< U4 > _P2PFuncs;`	Vector of P2P identifiers. This vector is initialized in an `init()` call with the id values of every P2P function in the trace file. This vector returns the related the corresponding collective operation id for every function id number.

`HashMap< U4, const char* > _communicatorsMapped;`	HashMap of all communicator names. Lookup is by id of communicator. Data value is strings.

3.3.5. Private Functions

Function	Explanation
`bool isSingleRankFunc( U4 _id_ )`	Checks to see whether the function id is a single rank ID. Returns TRUE if _id_ is a single rank function; FALSE otherwise.
`bool isCollectiveOperation( U4 _id_ )`	Checks to see whether the function id matches a collective operation ID. Returns TRUE if _id_ is a collective operation; FALSE otherwise.
`bool isP2P( U4 _id_ )`	This function checks to see whether the function id is a P2P ID. Returns TRUE if _id_ is a P2P function; FALSE otherwise.
`bool isAllToAll( U4 _id_ )`	Returns TRUE if it is an ALL-TO-ALL collective operation id number; FALSE otherwise.
`bool isOneToAll( U4 _id_ )`	Returns TRUE if it is a ONE-TO-ALL collective operation id number; FALSE otherwise.
`bool isAllToOne( U4 _id_ )`	Returns TRUE if it is an ALL-TO-ONE collective operation id number; FALSE otherwise.
`bool isAll( U4 _id_ )`	Returns TRUE if it is an ALL collective operation id number; FALSE otherwise.
`bool isSendP2P( U4 _id_ )`	Returns TRUE when the `FuncEventData` is a P2P send operation; FALSE otherwise.
`bool isSyncSendP2P( U4 _id_ )`	Returns TRUE when the `FuncEventData` is a P2P sync send operation; FALSE otherwise.
`bool isSendRecv( U4 _id_ )`	Returns TRUE when the `FuncEventData` is a P2P send- receive operation; FALSE otherwise.
`bool isRecv( U4 _id_ )`	Returns TRUE when the `FuncEventData` is a P2P receive operation; FALSE otherwise.

3.3.6. Typedef Functions

These functions are essential to providing the plug-in module for CPF. They are already included in `DevSim` and if you modified `DevSim`, then you should have properly modified these functions to handle your simulator. The `creat_t` function creates an instance of your simulator class and passes a reference back to CPF. When CPF has completed processing a trace, it destroys the instance of the simulator through the `destroy_t` function.

```
typedef CustomPluginFramework* create_t();
typedef void destroy_t(CustomPluginFramework*);
```

3.4. Filter Expression Grammar

The filter expression grammar creates a filter, which is a set of function (`funcfilter`), message (`p2pfilter`), and collective operation (`collfilter`) filters, each defining a filter for its respective kind of data. These sub-filter specifications are separated by the # sign and come in any order. Each filter class is specified either once, or more than once (in this case a Boolean AND is created from all subfilters for a given class), or not specified at all. An example where three classes of filters are specified is the expression generated in the graphical interface.

Each filter class specifier (`funcfilter`, `p2pfilter`, or `collfilter`) is followed by an expression put in parentheses. That expression can consist of any number of predicates that are different for each filter class and correspond to the entries described in the section Building Filter Expressions Using the Graphical Interface. These predicates are joined by using Boolean AND (&&) and OR (||) operators. Boolean expressions are parenthesized as needed. Also, a Boolean NOT (!) operator in front of any predicate or parenthesized expression negates the predicate/expression.

A filter class enables you to define a special expression to pass all or no data through the filter. For example, `p2pfilter(NONE)` filters out all messages, while `collfilter(ALL)` passes all collective operation. When the keywords ALL or NONE are used, ensure that it is the only argument to `funcfilter`, `p2pfilter`, or `collfilter`.

Keyword specification in the filter expression grammar is case-insensitive. Specifying names (for functions, processes and communicators, among others), however, is case-sensitive. Use double quotes for names that consist of several words or do not start with a letter or an underscore character (for example, "Major Function Groups"). Use double quotes for single word names (for example, "MPI") if necessary. White space (space and tab characters, as well as newlines) is ignored, unless it is part of a quoted name. If a process/group or function/group name is ambiguous, it is evaluated as if all matching groups were given.

3.4.1. Formal Description of Grammar

Here is a formal description of the filter expression grammar:

```
# The filter itself
FILTER ::= AFILTER
| FILTER # AFILTER
AFILTER ::= funcfilter ( FUNCFILTARG )
| collfilter ( COLLFILTARG )
| p2pfilter ( P2PFILTARG )
# Specifying functions
FUNCFILTARG ::= FUNCEXPR
| all
| none
FUNCEXPR ::= FUNCATOM
| FUNCEXPR && FUNCATOM
| FUNCEXPR || FUNCATOM
FUNCATOM ::= TG
| FG
| STARTTIME
| ( FUNCEXPR )
| ! FUNCATOM
# Specifying messages
P2PFILTARG ::= P2PEXPR
| all
| none
P2PEXPR ::= P2PATOM
| P2PEXPR && P2PATOM
| P2PEXPR || P2PATOM
P2PATOM ::= DURATION
| COMM
| TAG
| P2PVOLUME
| TGSENDER
| TGRECEIVER
| COMMSENDER
| COMMRECEIVER
| TGSRPAIR
| COMMSRPAIR
| TG
| COMMSR
| STARTTIME
| ENDTIME
| SENDER_FG
| RECEIVER_FG
| ( P2PEXPR )
| ! P2PATOM
# Specifying collective operations
COLLFILTARG ::= COLLEXPR
| all
| none
COLLEXPR ::= COLLATOM
| COLLEXPR && COLLATOM
| COLLEXPR || COLLATOM
COLLATOM ::= DURATION
| COMM
| COLLOPTYPE
| COLLVOLUME
| TGROOT
| COMMROOT
| TG
| STARTTIME
| ENDTIME
| ( COLLEXPR )
| ! COLLATOM
# Specifying times
STARTTIME ::= start ( TRIPLETS ; INTEGER )
| start ( TRIPLETS )
ENDTIME ::= end ( TRIPLETS ; INTEGER )
| end ( TRIPLETS )
DURATION ::= duration ( TRIPLETS )
# Specifying TGroups and FGroups
TG ::= tg ( NAMES )
FG ::= fg ( NAMES )
SENDER_FG ::= send_fg ( NAMES )
RECEIVER_FG ::= recv_fg ( NAMES )
# Specifying collective operation and message properties
COMM ::= comm ( TRIPLETS )
TAG ::= tag ( TRIPLETS )
COLLOPTYPE ::= type ( COLLNAMES )
COLLVOLUME ::= volume ( TRIPLETS )
P2PVOLUME ::= volume ( TRIPLETS )
# Specifying root, sender, or receiver, either by a TGroup name
# or by position in the communicator. If the operation has no
# root, then root() is always false.
TGROOT ::= root ( NAMES )
TGSENDER ::= sender ( NAMES )
TGRECEIVER ::= receiver ( NAMES )
# The predicate sr specifies both sender and receiver, separated
# by a semicolon.
TGSRPAIR ::= sr ( NAMES ; NAMES )
COMMROOT ::= root@ ( TRIPLETS )
COMMSENDER ::= sender@ ( TRIPLETS )
COMMRECEIVER ::= receiver@ ( TRIPLETS )
# The predicate sr@ specifies both sender and receiver ranks,
# separated by a semicolon.
COMMSRPAIR ::= sr@ ( TRIPLETS ; TRIPLETS )
```

```
COMMSR ::= tg@ ( TRIPLETS )
# Names containing fancy characters have to be double-quoted.
# Names map to TGroup and thread names, FGroup and function
# names, or collective operation types, depending on the context.
NAMES ::= NAME
| TRIPLET
| NAMES , NAME
| NAMES , TRIPLET
COLLNAMES ::= COLLNAMELIST
| TRIPLETS
COLLNAMELIST ::= NAME
| COLLNAMELIST , NAME
NAME ::= [_a-zA-Z][_a-zA-Z0-9.]*
| \"[^\"]*\"
# Specifying triplets and numbers
TRIPLETS ::= TRIPLET
| TRIPLETS , TRIPLET
TRIPLET ::= INTEGER
| INTEGER :
| INTEGER : INTEGER
| INTEGER : INTEGER : INTEGER
INTEGER ::= [0-9]+
```

3.4.2. Examples of Advanced Usage of Grammar

This section includes several examples of the manual usage of the filter expression grammar and how the manual usage of the grammar provides advanced capabilities in filtering trace data and speeding up the process of selecting exactly what you want to analyze.

For the first example, consider a parenthesized structure that cannot be built in the graphical interface easily (messages sent by process 0 and starting or ending between 70000 and 80000 ticks):

```
p2pfilter( sender( 0 ) && ( start( 70000:80000 ) || end( 70000:80000 ) ) )
```

The following example uses the `sr` predicate, which is not available in the graphical interface. This predicate helps to efficiently filter out all messages between processes 0 and 1:

```
p2pfilter( ! sr( 0:1; 0:1 ) )
```

Finally, consider the following scenario. With the graphical interface, a complicated filter is specified for a certain filter class with a large number of predicates and Boolean operators (both AND and OR, the latter added by using the **Add New Clause** button). To negate everything that has been specified so far (that is, to get exactly the trace data that was previously being filtered out), use `!` in front of the whole expression when in the manual mode. For example, the filter below specifies `MPI_Barrier` collective operations that last no longer than 2000 ticks, plus all collective operations with process 0 as the root:

```
collfilter( type( MPI_Barrier ) && duration( 0:2000 ) || root( 0 ) )
```

while the following filter specifies all the collective operations that do not match the description above:

```
collfilter( ! ( type( MPI_Barrier ) && duration( 0:2000 ) || root( 0 ) ) )
```