Intel® Trace Analyzer 6.0.1 User's Reference Guide

Intel® Corporation

Software and Solutions Group (SSG)

Developer Products Division (DPD)

Intel® Trace Analyzer 6.0.1 User's Reference Guide
by Intel® Corporation, Software and Solutions Group (SSG), Developer Products Division (DPD)

Table of Contents

Disclaimer and Legal Information

Notes

1. http://www.openjade.org/

Chapter 1. Introduction

The Intel® Trace Analyzer is a graphical tool that displays and analyzes event trace data generated by the Intel® Trace Collector. It helps in detecting performance problems, programming errors or in understanding the behavior of the application. This document describes the feature set of the Intel® Trace Analyzer.

1.1. Notation and Terms

- Menus and menu entries are printed as shown below:

  Main Menu⟶File⟶Exit

  This denotes the "Exit" entry in the "File" section of the Main Menu.

- Shell commands are printed with a leading '$'. For example,

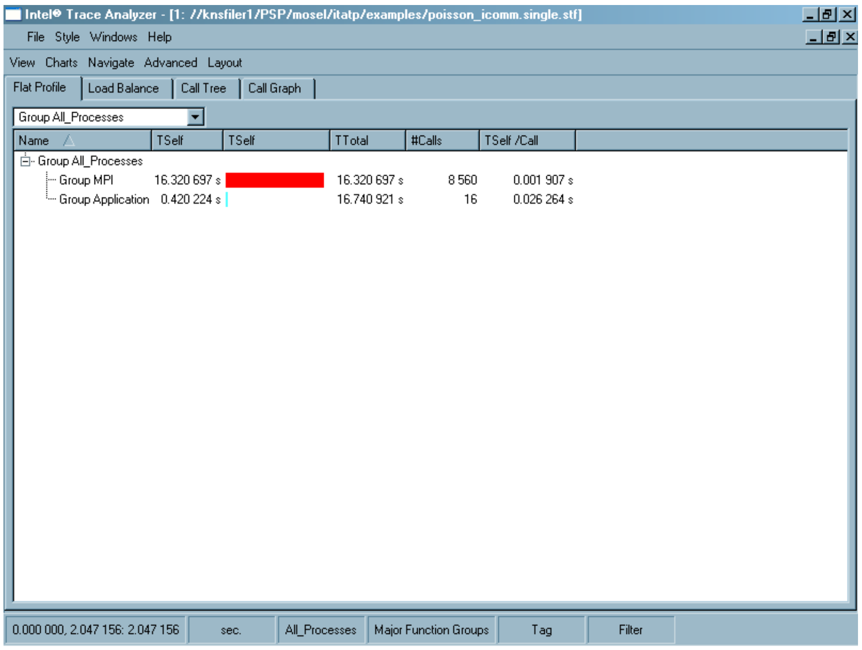
  ```
  $ ls
  ```

  denotes the UNIX command "ls".

- The following line is an example of how source code is presented:

  ```
  CALL MPI_FINALIZE()
  ```

The term "process" in this documentation implicitly includes threads. As soon as the Intel® Trace Analyzer loads a trace file that was generated running a multi-threaded application, the GUI uses the term "thread" wherever applicable. This is done to avoid confusing MPI application programmers who do not use threads with this term.



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group MPI	16.320 697 s	🟥	16.320 697 s	8 560	0.001 907 s
Group Application	0.420 224 s		16.740 921 s	16	0.026 264 s

Figure 1-1. The Intel® Trace Analyzer

1.2. Starting Intel® Trace Analyzer

Under UNIX, invoke the Intel® Trace Analyzer 6.0 via the command line by typing

```
$ traceanalyzer
```

Assuming that the executable is found, this command launches the Intel® Trace Analyzer. Optionally, one or more trace files are specified as arguments as shown in the following example:

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

The above example opens the trace file `poisson_icomm.single.stf` in the Intel® Trace Analyzer. To open trace files within the Intel® Trace Analyzer without restarting it, use the File Menu entry **Main Menu**⟶**File**⟶**Open**. Either way, each time a trace file is opened, a new window displaying the trace file's Function Profile Chart is displayed(See Figure 1-2.

> **Note:** A file opened in the Intel® Trace Analyzer should not be changed. In such a case, its proper functioning cannot be guaranteed.

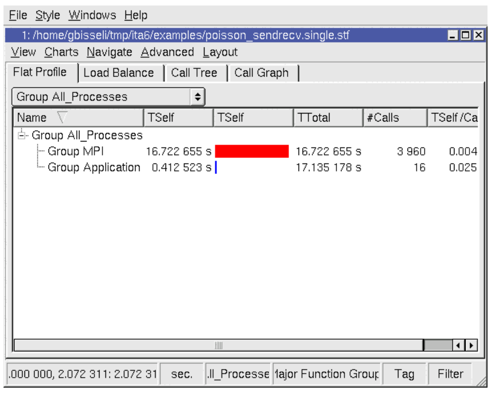
In a Windows* environment, click on a trace file to invoke the Intel® Trace Analyzer or use the menu item **Start**⟶**All Programs**⟶**Intel® Trace Analyzer**.

1.3. For the Impatient

This section provides a quick start into the Intel®Trace Analyzer. It demonstrates essential features of the program using the example trace files `poisson_sendrecv.single.stf` and `poisson_icomm.single.stf` that are available in the Intel® Trace Analyzer's examples directory.

The traces were generated with two implementations of the same algorithm computing the same result over the same data set: a poisson solver for a linear equation system. As the names imply, the first version uses a sendrecv to communicate, while the second version uses non-blocking communication.

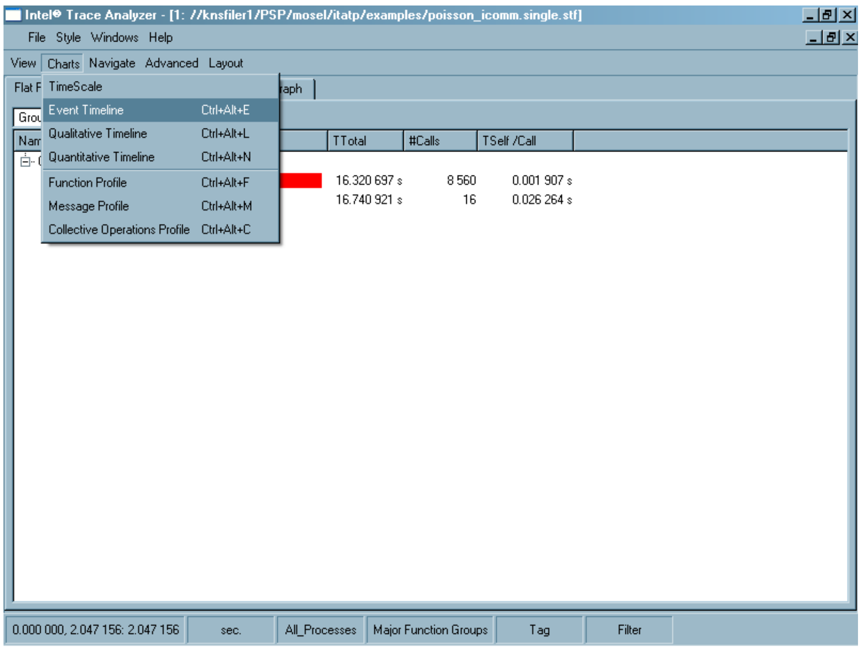
It is illustrated that the first version leads to an overall serialization of the parallel algorithm and how the improved version solves the problem. Figure 1-2 shows Intel®Trace Analyzer after loading `poisson_sendrecv.single.stf`. The figure shows a main window and a child window, a so called *View*. The View contains a function profile which shows that the program spent nearly all of its time in MPI.



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Ca
Group All_Processes					
Group MPI	16.722 655 s	🟥	16.722 655 s	3 960	0.004
Group Application	0.412 523 s		17.135 178 s	16	0.025

Figure 1-2. poisson_sendrecv.single.stf loaded

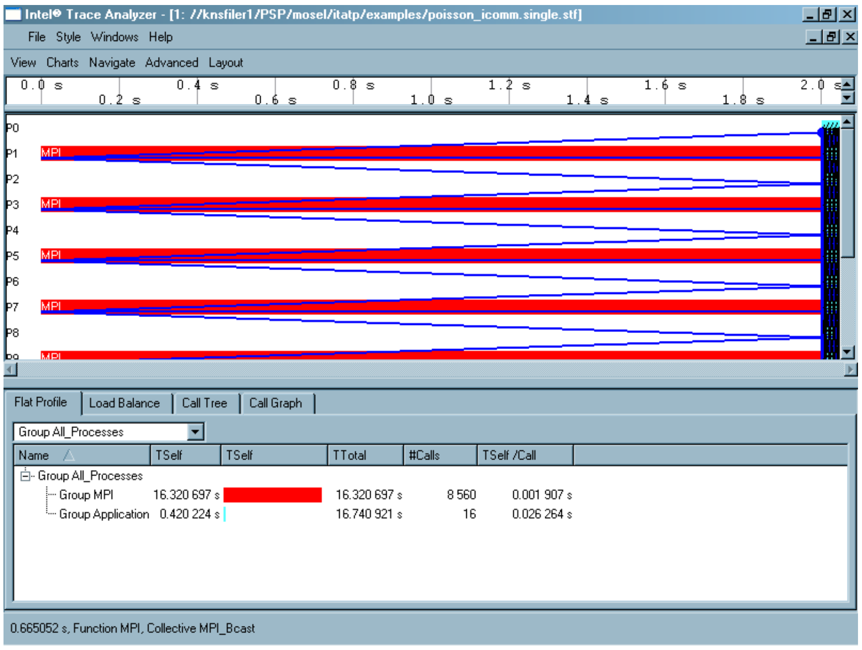
Maximize the View with the respective button in its title bar and open an Event Timeline (**View Menu**⟶**Charts**⟶**Event Timeline**) as shown in Figure 1-3. The result should look like in Figure 1-4.


Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
TimeScale
Event Timeline Ctrl+Alt+E
Qualitative Timeline Ctrl+Alt+L
Quantitative Timeline Ctrl+Alt+N
Function Profile Ctrl+Alt+F
Message Profile Ctrl+Alt+M
Collective Operations Profile Ctrl+Alt+C

TTotal	#Calls	TSelf /Call
16.320 697 s	8 560	0.001 907 s
16.740 921 s	16	0.026 264 s

0.000 000, 2.047 156: 2.047 156 | sec. | All_Processes | Major Function Groups | Tag | Filter

Figure 1-3. Opening an Event Timeline



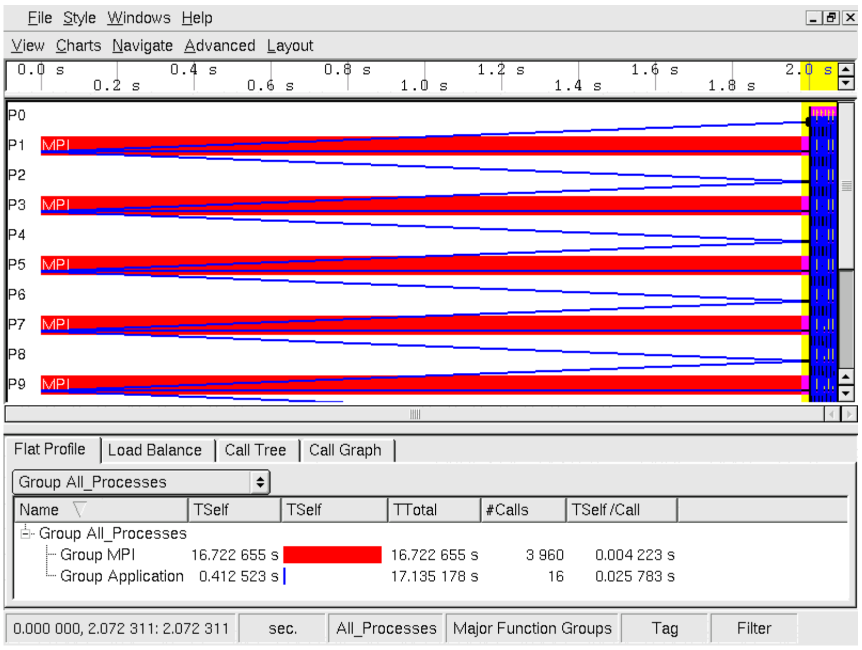
Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group MPI	16.320 697 s	🟥	16.320 697 s	8 560	0.001 907 s
Group Application	0.420 224 s		16.740 921 s	16	0.026 264 s

0.665052 s, Function MPI, Collective MPI_Bcast

Figure 1-4. Event Timeline opened

Together with the Event Timeline (see Section 4.1), a time scale opens above the Timeline in the View. Figure 1-4 shows a View containing a time scale, an Event Timeline and a Function Profile. These diagrams are called Charts (see Chapter 4). In the status bar (see Section 3.2) found at the bottom of the View, At the bottom of the View in the status bar, the current time interval and some other information are shown.

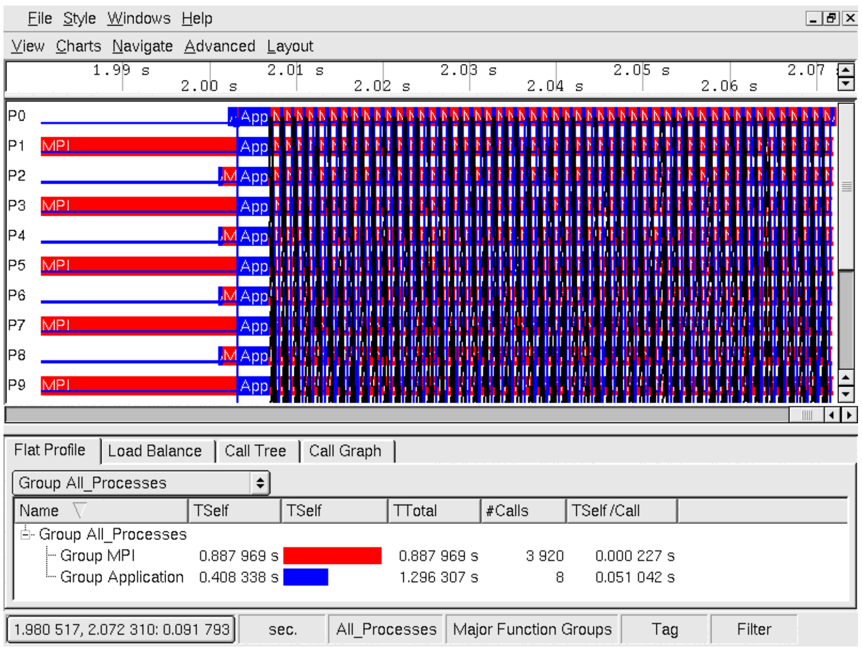
The setup of the parallel processes seems to take most of the run time. Zoom into the interesting area on the right edge of the Event Timeline by dragging the mouse over the desired time interval with the left mouse button pressed as shown in Figure 1-5. The result should look like in Figure 1-6. Note the apparent iterative nature of the application.



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group MPI	16.722 655 s	🟥	16.722 655 s	3 960	0.004 223 s
Group Application	0.412 523 s	🟦	17.135 178 s	16	0.025 783 s

0.000 000, 2.072 311: 2.072 311

Figure 1-5. Zooming into a Chart



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group MPI	0.887 969 s	🟥	0.887 969 s	3 920	0.000 227 s
Group Application	0.408 338 s	🟦	1.296 307 s	8	0.051 042 s

1.980 517, 2.072 310: 0.091 793

Figure 1-6. Zoomed result

Now zoom further into the trace to look at a single iteration and close the Function Profile (**Context Menu**⟶**Close Chart**). The result should look like Figure 1-7.

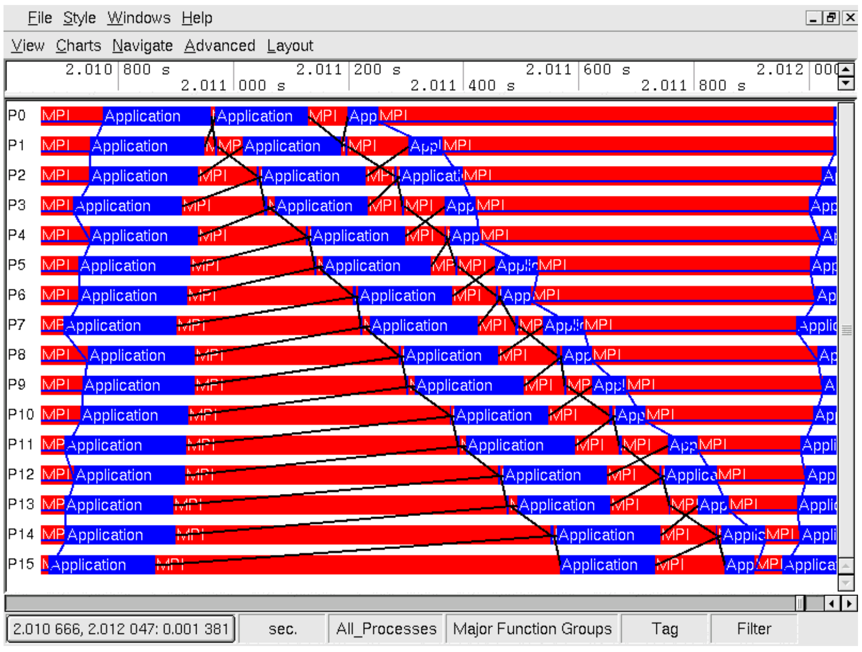


Figure 1-7. Zoomed to one iteration

To see which particular MPI functions are used in the program, right-click on MPI in the Event Timeline and choose *Ungroup Group MPI* shows this. The result should look like Figure 1-8. The Function Aggregation of the View changes so that the MPI functions are no longer aggregated (see Section 7.2) into the Function Group MPI but are shown separately. This is shown in the status bar. The button titled *Major Function Groups* changes to *MPI expanded in (Major Function Groups)*. Click this button to open the *Function Group Editor* (see Section 5.4), which enables creating new function groups and to switch between them.

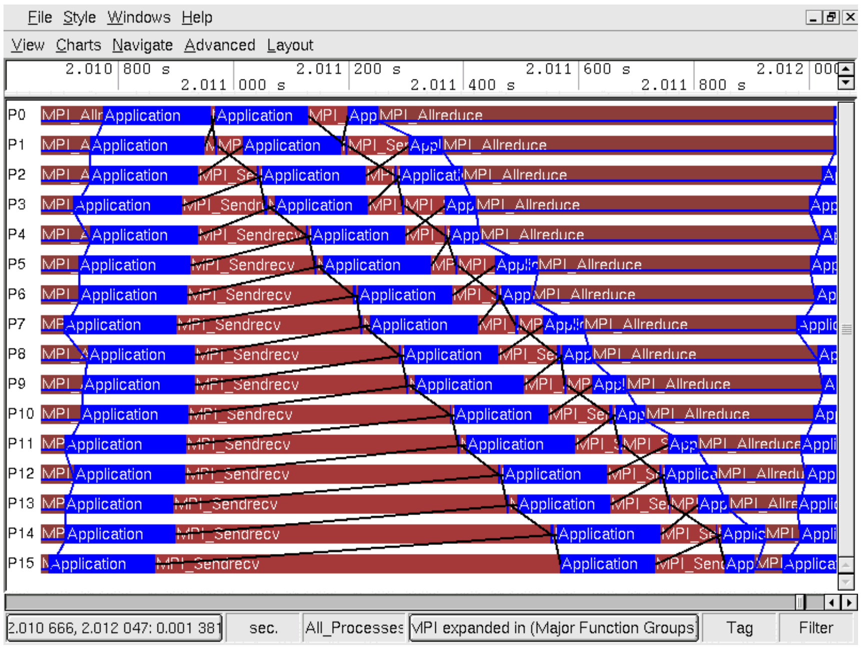


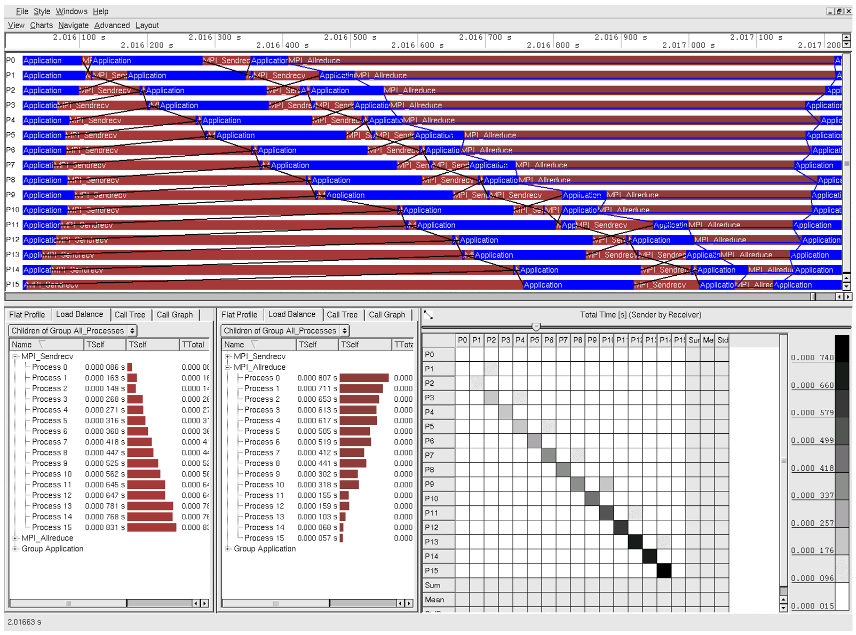
Figure 1-8. MPI ungrouped

It is apparent that at the start of the iteration the processes communicate with their direct neighbors using `MPI_Sendrecv`. The way this data exchange is implemented shows a clear disadvantage: process *i* does not exchange data with its neighbor *i+1* until the exchange between *i-1* and *i* is complete. This makes the first `MPI_Sendrecv` block look like a staircase. The second block is already deferred and hence does not show the same effect. The `MPI_Allreduce` at the end of the iteration nearly resynchronizes all processes. The net effect is that the processes spend most of their time waiting for each other.

Looking at the status bar (see Figure 1-8) shows that one iteration is roughly 1.4 milliseconds long.

Figure 1-9 shows a View with an Event Timeline (**View Menu**⟶**Charts**⟶**Event Timeline**), two Function Profiles (**View Menu**⟶**Charts**⟶**Function Profile**) with their Load Balance tab and a Message profile (**View Menu**⟶**Charts**⟶**Message Profile**) that reveals the asymmetric pattern in the point to point messages.

Note that the time spent in `MPI_Sendrecv` grows with the process number while the time for `MPI_Allreduce` decreases. The Message Profile (Section 4.5) in the bottom right corner of Figure 1-9 shows that messages traveling from a higher rank to a lower rank need more and more time with increasing rank while the messages traveling from lower rank to higher rank do reveal a weak even-odd kind of pattern.


File Style Windows Help
View Charts Navigate Advanced Layout
Flat Profile
Load Balance
Call Tree
Call Graph
Children of Group All_Processes
Total Time [s] (Sender by Receiver)
2.01663 s

```
Name          TSelf              TTotal
MPI_Sendrecv
  Process 0   0.000 086 s        0.000 08
  Process 1   0.000 163 s        0.000 16
  Process 2   0.000 149 s        0.000 14
  Process 3   0.000 268 s        0.000 26
  Process 4   0.000 271 s        0.000 27
  Process 5   0.000 316 s        0.000 31
  Process 6   0.000 360 s        0.000 36
  Process 7   0.000 418 s        0.000 41
  Process 8   0.000 447 s        0.000 44
  Process 9   0.000 525 s        0.000 52
  Process 10  0.000 562 s        0.000 56
  Process 11  0.000 645 s        0.000 64
  Process 12  0.000 647 s        0.000 64
  Process 13  0.000 781 s        0.000 78
  Process 14  0.000 768 s        0.000 76
  Process 15  0.000 831 s        0.000 83
MPI_Allreduce
Group Application
```

```
Name          TSelf              TTota
MPI_Sendrecv
MPI_Allreduce
  Process 0   0.000 807 s        0.000
  Process 1   0.000 711 s        0.000
  Process 2   0.000 653 s        0.000
  Process 3   0.000 613 s        0.000
  Process 4   0.000 617 s        0.000
  Process 5   0.000 505 s        0.000
  Process 6   0.000 519 s        0.000
  Process 7   0.000 412 s        0.000
  Process 8   0.000 441 s        0.000
  Process 9   0.000 302 s        0.000
  Process 10  0.000 318 s        0.000
  Process 11  0.000 155 s        0.000
  Process 12  0.000 159 s        0.000
  Process 13  0.000 103 s        0.000
  Process 14  0.000 068 s        0.000
  Process 15  0.000 057 s        0.000
Group Application
```

Figure 1-9. Many Charts

As `poisson_sendrecv.single.stf` is such a striking example of serialization, next to all Charts provided by Intel® Trace Analyzer reveal this interesting pattern. But in real world cases it might be necessary to formulate a hypothesis regarding how the program should behave and to check this hypothesis using the most adequate Chart.

A possible way to improve the performance of the program is to use non-blocking communication to replace the usage of `MPI_Sendrecv` and to avoid the serialization in this way. One iteration of the resulting program looks like the one shown in Figure 1-10. Note that a single iteration now takes about 0.9 milliseconds, while it took about 1.4 milliseconds before the change.

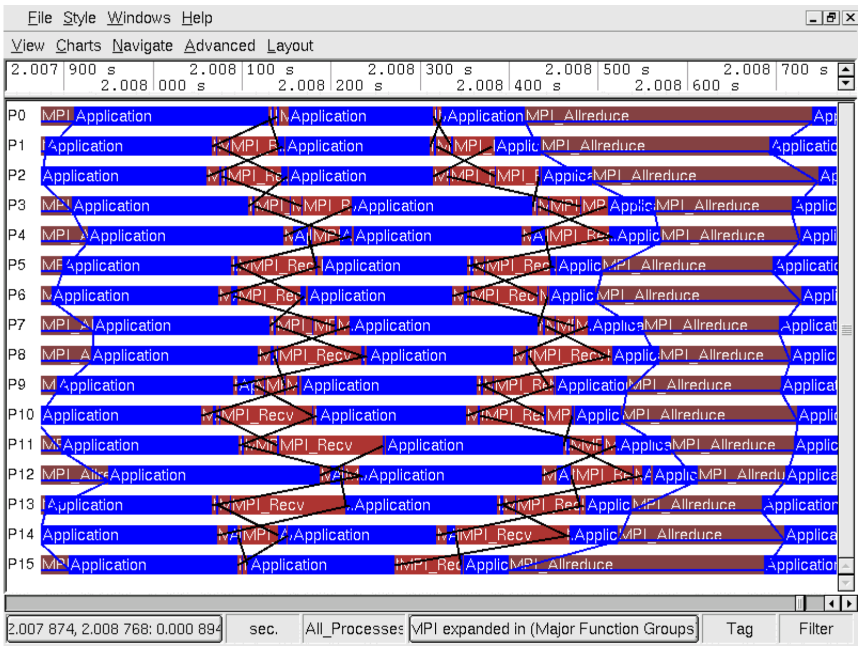


Figure 1-10. One iteration of the improved version

Of course this introduction only scratches the surface. If time permits, it is recommended to browse through the whole documentation. If time is tight, as it always is, it is recommended to at least read through Chapter 7 to whet the appetite for features like filtering, tagging, process aggregation and function aggregation. These features have the potential to make analyzing parallel applications more efficient.

Have fun!

Chapter 2. The Main Menu

The main menu of the Intel® Trace Analyzer contains options using which the general settings of the Analyzer can be changed. Use these entries to edit the configuration settings and the display style.

There are four sub-menus under the main menu. These are:

1. File Menu
2. Style Menu
3. Windows Menu
4. Help Menu

2.1. The File Menu

The File menu has four entries. The *Open* menu entry opens a dialog. Specify one or several trace files to open in this dialog.

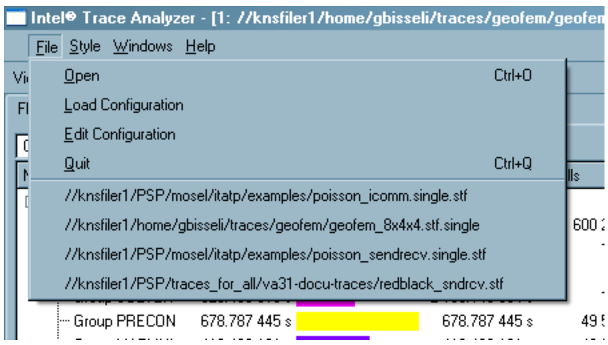

Intel® Trace Analyzer - [1: //knsfiler1/home/gbisseli/traces/geofem/geofen
File Style Windows Help
Open Ctrl+O
Load Configuration
Edit Configuration
Quit Ctrl+Q
//knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf
//knsfiler1/home/gbisseli/traces/geofem/geofem_8x4x4.stf.single
//knsfiler1/PSP/mosel/itatp/examples/poisson_sendrecv.single.stf
//knsfiler1/PSP/traces_for_all/va31-docu-traces/redblack_sndrcv.stf
Group PRECON 678.787 445 s 678.787 445 s

Figure 2-1. The File Menu

To select a new configuration, use the *Load Configuration* entry. This opens a dialog where the required configuration is selected.

The option *Edit Configuration* opens the Configuration dialog which is explained in Section 5.10.1.

The *Quit* entry exits the Intel® Trace Analyzer. Below the *Quit* entry, there is a list of the ten trace files opened most recently.

2.2. The Style Menu

The *Style* menu provides the option of choosing the design in which the view is displayed. The options under the *Style* menu depend on the environment in use. The first entry always shows the default style for the respective platform (or for the respective window manager when using X Windows). For example, under a Windows environment, the user has the option of choosing between the *QWindowsXPStyle* option, the *Windows* option and the *WindowsXP* option. In a KDE environment,the user has the default as *QMotifStyle* and *CDE, Motif, MotifPlus, Platinum, SGI* and *Windows* as other options.

Additional styles may be available when provided by a desktop environment like KDE. If you experience problems with those styles then switch back to the default style. There will be no support in case of problems that can not be reproduced using the default style.

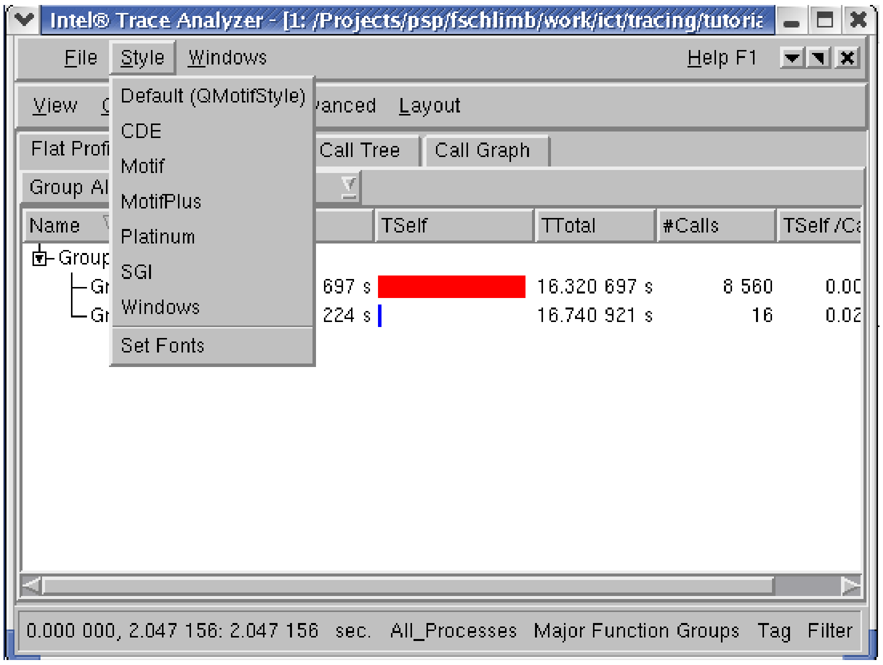


Figure 2-2. The Style Menu

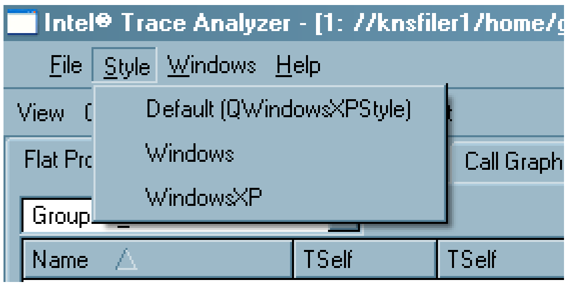


Figure 2-3. The Style Menu under Windows

2.3. The Windows Menu

Use the menu options in the *Windows* menu to arrange open sub-windows as required. There are three possibilities which are explained below. The *Windows* sub-menu also shows the name and path of the trace file that is presently open.

- *Cascade*

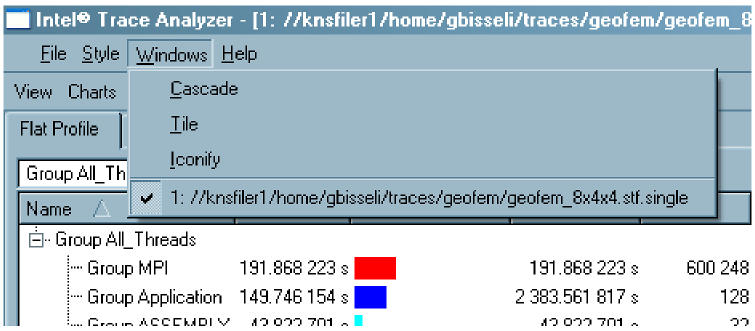
  Select the *Cascade* option to arrange the open sub-windows one behind the other.

- *Tile*

  The *Tile* option arranges the sub-windows next to each other. Note that this option is useful only when charts are opened in two or more sub-windows.

- *Iconify*

  Iconify minimizes the open sub-windows and shows the icon of these in the bottom of the View.


Intel® Trace Analyzer - [1: //knsfiler1/home/gbisseli/traces/geofem/geofem_8
File Style Windows Help
View Charts
Flat Profile
Group All_Th
Name
Cascade
Tile
Iconify
1: //knsfiler1/home/gbisseli/traces/geofem/geofem_8x4x4.stf.single

```
Group All_Threads
   Group MPI          191.868 223 s      191.868 223 s    600 248
   Group Application  149.746 154 s    2 383.561 817 s        128
```

Figure 2-4. The Windows Menu

2.4. The Help Menu

The Help menu give you details regarding the Intel® Trace Analyzer in use. Also, on pressing F1 when in any of the Charts, context menus or dialog boxes, the context sensitive help tool opens giving a description of the current feature.

Chapter 3. Views

For a flexible analysis of a trace file it is helpful to look at multiple partitions of the data from various perspectives. The concept of Views is introduced here. A View holds a collection of Charts in a single window. These Charts, inherent in the same View, use the same perspective on the data. This perspective is made up of the following attributes: the time interval, process aggregation, function aggregation and filters. For an in-depth explanation of aggregation and filters, refer to Chapter 7

Whenever an attribute in the current perspective is changed for one of the Charts, all other Charts follow. Opening several Views offers a very flexible and variable mechanism for exploring, analyzing and comparing trace data. The View's status bar shows the current perspective on the data and allows to quickly change the perspective. More about the status bar is described in Section 3.2.

3.1. The View's Main Menu

All operations that can be performed on a View are found on the View's menu bar. This menu bar consists of five menu options:

- *View Menu*
- *Charts Menu*
- *Navigate Menu*
- *Advanced Menu*
- *Layout Menu*

3.1.1. The View Menu

The View menu consists of options that are carried out on the entire View. These options are described below:

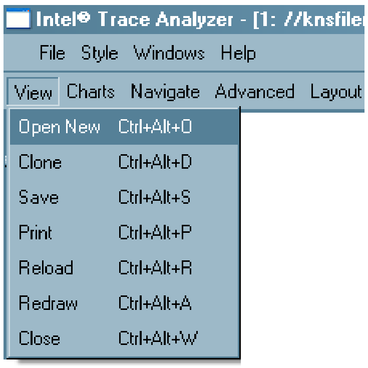


Figure 3-1. The View Menu

- *Open New (Ctrl+Alt+O)*

  This opens the *New View* dialog box, which opens a new View with the selected Charts. The *New View* dialog box is explained in Section 5.9.

- *Clone (Ctrl+Alt+D)*

  This command replicates the existing View. It creates a one-to-one clone of the current View.

- *Save (Ctrl+Alt+S)*

  This command saves the entire View as a picture.

- *Print (Ctrl+Alt+P)*

  This entry prints a copy of the View.

- *Redraw (Ctrl+Alt+R)*

  This command repaints the entire View.

- *Close (Ctrl+Alt+W)*

  This command closes the current View. If the last View for a trace file is closed, the trace file is closed as well.

- *Compare*

  This opens a dialog to select another already opened trace file which can be just the same as the current one. After choosing a new Comparison View will be opened that displays data from the two files. See Chapter 6.

3.1.2. The Charts Menu

The Charts menu contains entries to open the various Charts of the Intel® Trace Analyzer. There are seven entries in this menu:

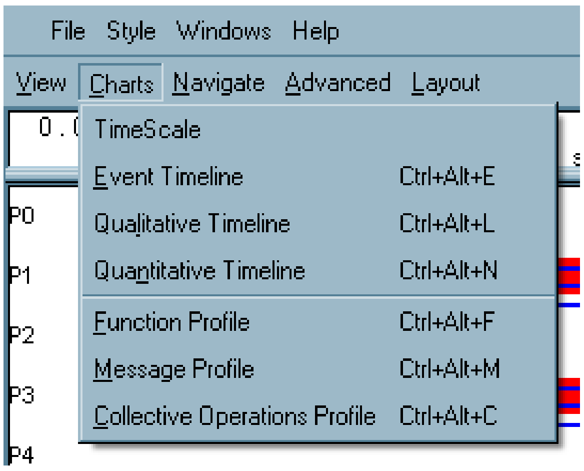

File
Style
Windows
Help
View
Charts
Navigate
Advanced
Layout
TimeScale
Event Timeline Ctrl+Alt+E
Qualitative Timeline Ctrl+Alt+L
Quantitative Timeline Ctrl+Alt+N
Function Profile Ctrl+Alt+F
Message Profile Ctrl+Alt+M
Collective Operations Profile Ctrl+Alt+C
P0
P1
P2
P3
P4

Figure 3-2. The Charts Menu

- *TimeScale*

  This TimeScale entry displays the Timescale. The Timescale also appears automatically on first opening of any Timeline Chart.

- *Event Timeline (Ctrl+Alt+E)*

  The Event Timeline is a Chart that visualizes what the individual processes are doing. The Event Timeline is explained in detail in Section 4.1.

- *Qualitative Timeline (Ctrl+Alt+L)*

The Qualitative Timeline shows event attributes as they occur over time. This timeline is explained in Section 4.2.

- *Quantitative Timeline (Ctrl+Alt+N)*

  This entry opens the Quantitative Timeline. It gives an overview of the parallel behavior of the application and is explained in Section 4.3.

- *Function Profile (Ctrl+Alt+F)*

  This entry opens the Function Profile. On opening a trace file in the Intel® Trace Analyzer, the Function Profile is shown by default. For more information on the Function Profile, see Section 4.4.

- *Message Profile (Ctrl+Alt+M)*

  The Message Profile provides statistical information on MPI point to point messages. Messages are categorized by grouping them in two dimensions. A detailed description of the Message Profile is available under Section 4.5.

- *Collective Operations Profile (Ctrl+Alt+C)*

  The Collective Operations Profile provides statistical information on MPI's collective operations. Operations are categorized by grouping them in two dimensions. For more information on the Collective Operations Profile, refer to Section 4.6.

3.1.3. The Navigate Menu

All functions provided by the Navigation Menu are also available through convenient keyboard short cuts. The entries in the Navigate menu are:

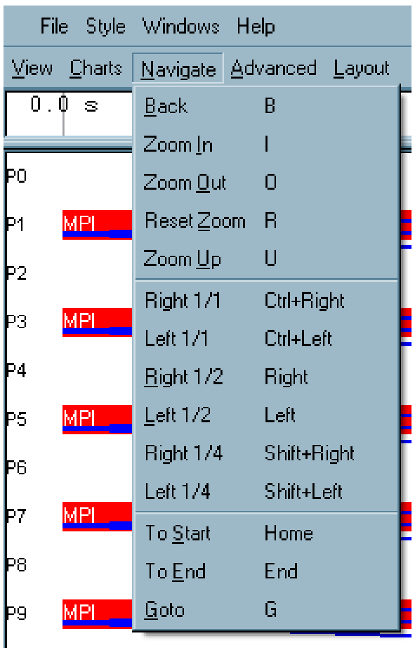


Figure 3-3. The Navigate Menu

- *Back (B):*

  Pops the current interval from the zoom stack (Section 3.3.1).

- *Zoom In (I):*

  Magnifies by a factor of two around the current center and pushes this new interval onto the zoom stack (Section 3.3.1).

- *Zoom Out (O):*

  Scales by a factor of 1/2 around the current center and pushes this new interval onto the zoom stack (Section 3.3.1).

- *Reset Zoom (R):*

  Clears the zoom stack and then pushes a time interval onto the zoom stack (Section 3.3.1) that corresponds to the complete trace file.

- *Zoom Up (U):*

  Maintains the current center and goes back to the previous zoom step.

- *Left 1/1 (Ctrl-Left):*

  Moves one whole screen to the left and pushes the new interval onto the zoom stack (Section 3.3.1).

- *Right 1/1 (Ctrl-Right):*

  Moves one whole screen to the right and pushes the new interval onto the zoom stack (Section 3.3.1).

- *Left 1/2 (Left):*

  Moves half a screen to the left and pushes the new interval onto the zoom stack (Section 3.3.1).

- *Right 1/2 (Right):*

  Moves half a screen to the right and pushes the new interval onto the zoom stack (Section 3.3.1).

- *Left 1/4 (Shift-Left):*

  Moves a quarter of a screen to the left and pushes the new interval onto the zoom stack (Section 3.3.1).

- *Right 1/4 (Shift-Right):*

  Moves a quarter of a screen to the right and pushes the new interval onto the zoom stack (Section 3.3.1).

- *To Start (Home):*

  Moves to the start of the trace file and pushes the new interval onto the zoom stack. (Section 3.3.1)

- *Top End (End):*

Moves to the end of the trace file and pushes the new interval onto the zoom stack (Section 3.3.1).

- *Goto (G):*

  Opens the *Time Interval Selection* dialog box (Section 5.8).

3.1.4. The Advanced Menu

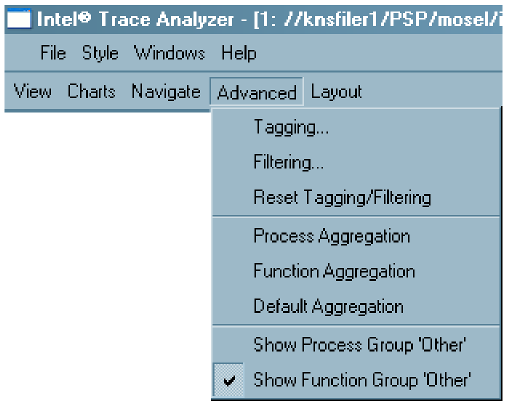

Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/i
File Style Windows Help
View Charts Navigate Advanced Layout
Tagging...
Filtering...
Reset Tagging/Filtering
Process Aggregation
Function Aggregation
Default Aggregation
Show Process Group 'Other'
Show Function Group 'Other'

Figure 3-4. The Advanced Menu

- *Tagging*

  Tagging highlights events that satisfy specific user-defined conditions. Click on this entry to open the dialog box where the tagging conditions are specified. For more information on the Tagging dialog box. refer to Section 5.2.

- *Filtering*

  Filtering shows only those events that satisfy a condition as defined by the user. All other events are suppressed as if they had never been written to the trace file. The filtering dialog box is explained in Section 5.1.

- *Reset Tagging/Filtering*

  This menu entry is active only when the chart has been either filtered or tagged. It undoes the tagging/filtering. All events are shown and none are filtered or tagged.

- *Process Aggregation*

  Process Aggregation focusses on the processes that are of importance to the user and aggregates the results to Process Groups. It accesses the Process Group Editor (Section 5.3) where the appropriate process groups can be constructed and/or chosen.

- *Function Aggregation*

  Function Aggregation focuses on a subset of functions and aggregates these into Function Groups, without being distracted by other functions that are currently not significant to the user. Use the To do the Function Group Editor (Section 5.4) to do this.

- *Default Aggregation*

  Default aggregation sets the aggregation values back to the default settings which are *All Processes* for process aggregation and *Major Function Groups* for function aggregation.

- *Show Process Group "Other"*

  If the process aggregation does not cover all processes present in the trace file then all data for uncovered processes are subsumed in the artificially created process group *Other*.

  Use this switch to show or hide the process group *Other*. The default is to hide this group.

  The Quantitative Timeline and the Function Profile enable selecting a subset of the current processes to be shown and can hence override this switch.

- *Show Function Group "Other"*

  If the current function aggregation does not cover all functions in the trace file then all uncovered function events are subsumed in the artificially created function group *Other*.

  Use this switch to show or hide this group. By default, the switch is enabled to show this group.

  The Event Timeline always shows the Function Group *Other* wherever applicable instead of showing "holes". The Quantitative Timeline and the Function Profile enable selecting a subset of the current functions to be shown and can therefore override this switch.

3.1.5. The Layout Menu

The Layout menu adjusts the position of the Chart on screen. For example, it provides the option of viewing the timelines and profiles next to each other or like the default, one below the other. The different entries of the layout menu are explained below:

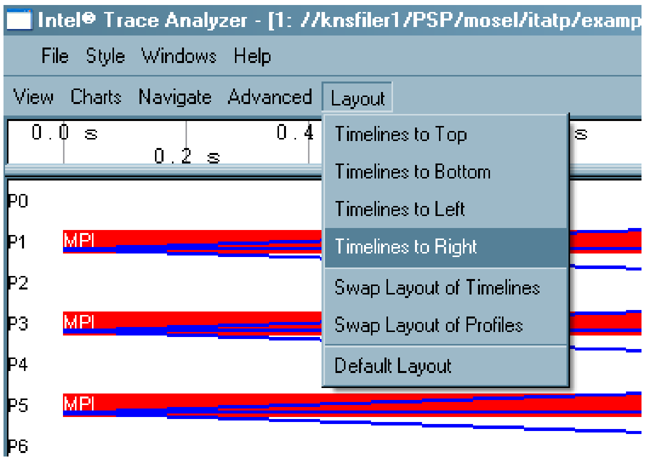


Figure 3-5. The Layout Menu

- *Timelines to Top*

Use this entry to move the timelines to the upper portion of the View and the profiles to the bottom.

- *Timelines to Bottom*

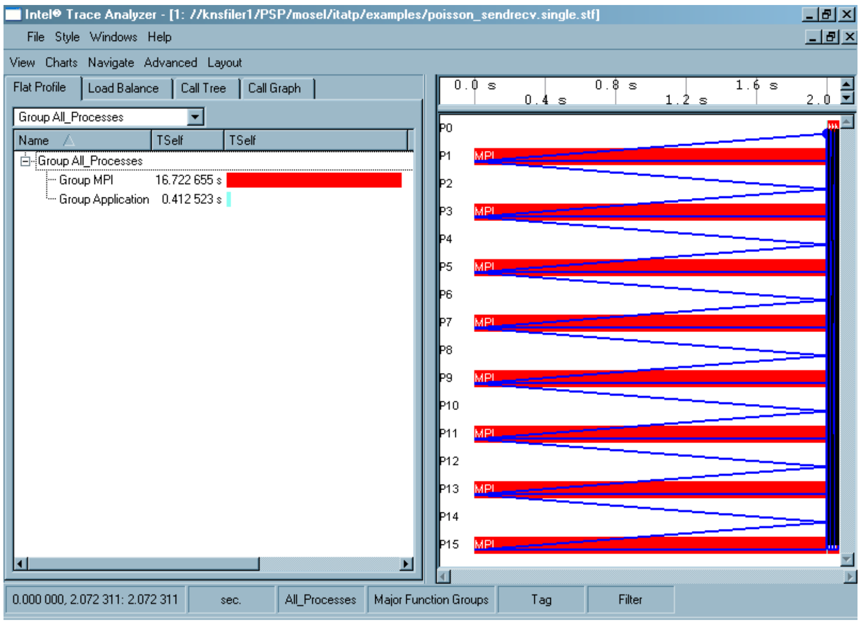
  Use this entry to move the timelines to the lower portion of the View and the profiles to the top.

- *Timelines to Left*

  Use this entry to place the timelines to the left of the View. For example, to display the Charts next to each other, select the option **View Menu**⟶**Layout**⟶**Timelines to Left**. This moves the timeline(s) being shown in the View to the left of the screen, and thereby also shifts the profiles being shown to the other side. In case multiple timelines are open, these are collectively shown one below the other on the chosen side.

- *Timelines to Right*

  Use this entry to place all the timelines to the right of the View. It works the same way as *Timelines to the Left*.



Name	TSelf	TSelf
Group All_Processes		
Group MPI	16.722 655 s	🟥
Group Application	0.412 523 s	

Figure 3-6. Placing timelines to the right

- *Swap Layout of Timelines*

  Swaps the position 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.

- *Swap Layout of Profiles*

  Swaps the position of the profiles relative to each other, similar to that of the timelines as explained above.

- *Default Layout*

This option brings the layout of the View back to the default settings. By default, timelines are stacked in the upper portion of the View and Profiles in the lower portion, side by side.

3.2. The Status Bar

When moving the mouse cursor over an event in a timeline or an entry in a profile, expanded information on the respective entry is seen in the status bar at the bottom of the View for a few seconds. After this, or when the mouse moves into the status bar, it provides buttons that show the current settings and that enable changing the current settings. These shortcut buttons are described below:

- *Time Interval*

  The leftmost button in the status bar opens a dialog box that sets the time interval. More about the time interval dialog box is available in Section 5.8. The title on this button is of the form “$Start, $End: $Duration”. The status bar in Figure 3-6 shows that the start position of the time interval is at 0.0 seconds, the end position at 2.072 seconds and that the duration is 2.072 seconds. The button appears flat when the full time range of the file is shown.

- *Sec./Tick*

  This button shows the current unit of time and switches the unit from seconds to ticks and vice versa. The button appears flat if the unit in use is seconds and appears raised when using ticks.

- *Process Aggregation*

  This button shows the current thread aggregation. By default, this is *All_Processes*. Click this button to open the *Process Group Editor* dialog box, which is explained in Section 5.3. The button appears flat for the default thread aggregation.

- *Function Aggregation*

  This button shows the current function aggregation. Per default, this is *Major Function Groups*. Clicking this button opens the *Function Group Editor* dialog box, which is explained in Section 5.4. The button appears flat for the default function aggregation.

- *Tagging and Filtering*

  The buttons labeled *Tag* and *Filter* open the *Tagging* dialog box (see Section 5.2) and the *Filtering* dialog box (see Section 5.1) respectively. These buttons appear flat if no tagging/filtering is selected.

3.3. Navigation In Time

In addition to the menu entries and keystrokes mentioned in Section 3.1.3, there are two more ways to move in time in the Intel® Trace Analyzer. One is by using the scroll bar found below the timelines and another is by using the mouse to zoom into a time interval. Similar to the keystrokes, these operations manipulate the zoom stack. Refer to Section 3.3.1 for a detailed explanation.

To move the View by a quarter of a screen to a selected direction, click on 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 into the timeline (even clicking into the Timescale 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.

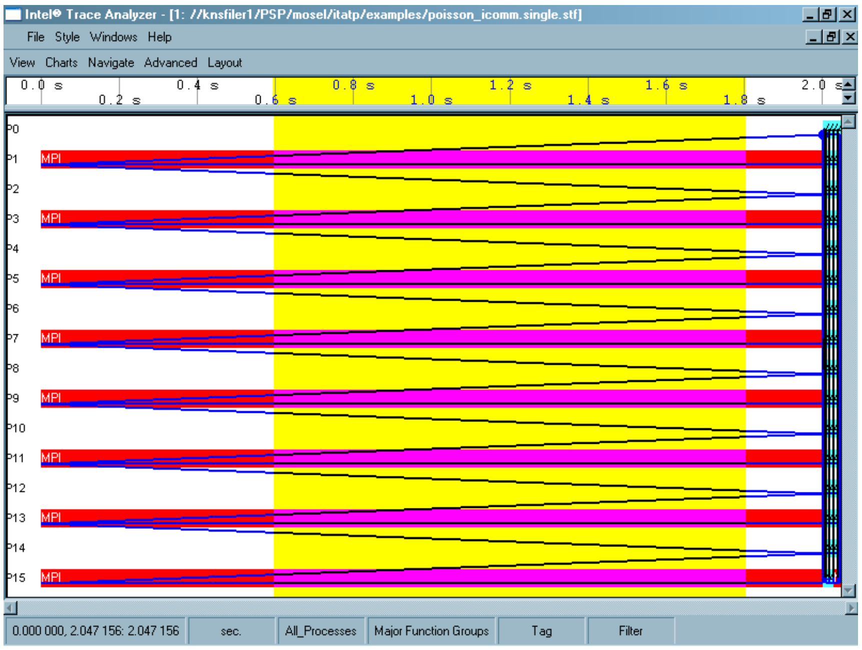


Figure 3-7. Zooming with the mouse

3.3.1. The Zoom Stack

The zoom stack supports navigation in time by storing previously displayed time intervals. These time intervals are restored when required. Navigation using the keyboard or mouse is quite intuitive without detailled 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)
```

Figure 3-8. The zoom stack

Each stack entry has the syntax *(Start; Stop)(Center; Width)*. Zooming in using *I (Zoom In)*, 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 *I* will result in a stack as shown in Figure 3-9.

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

Figure 3-9. State of the zoom stack - Zoomed in twice

Figure 3-9 shows 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. Using the state shown in Figure 3-10, the differences between *Back (B)*, *Zoom Out (O)* and *Zoom Up (U)* are explained below.

```
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)
```

Figure 3-10. State of the zoom stack - Moved two window sizes to the right

Back (B): Pressing B in the state shown in Figure 3-10 pops out one level of the stack and the previous time frame is displayed as shown in Figure 3-11.

```
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)
```

Figure 3-11. State of the zoom stack - after *Back (B)*

Zoom Out: Using *O* in Figure 3-10 lowers the magnification (doubles the width) and pushes a new item on the stack. Zooming out using *O* in the state shown in Figure 3-10 results in the stack as shown below.

```
6:(70;90)(80;20)
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)
```

Figure 3-12. State of the zoom stack - after *Zoom Out (O)*

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 Figure 3-10) and to clear the stack up to and including the last "zoom in" found. In our example the current center is 80 and the width from stack level 2, which is 20, is used.

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

Figure 3-13. State of the zoom stack - after *Zoom Up (U)*

Reset (R) : *Reset* clears the stack and pushes one entry covering the full time range of the trace file. Therefore, on pressing *R*, the stack shown in Figure 3-8 is reestablished.

Chapter 4. Charts

Charts in the Intel® Trace Analyzer are graphical or alphanumerical diagrams that are parameterized with a time interval, a process grouping and a function grouping (see Section 7.2) 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 the Intel® Trace Analyzer is divided into:

1. *Timelines:* the Event Timeline, the Qualitative Timeline and the Quantitative Timeline.
2. *Profiles:* the Function Profile, the Message Profile and the Collective Operations Profile.

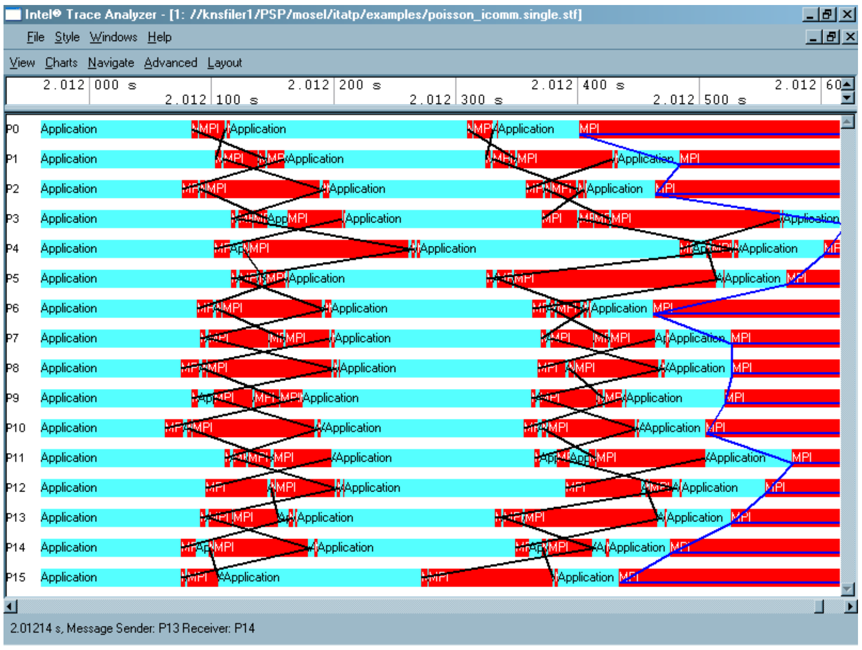
Charts are grouped into Views. These Views provide ways to choose the time interval, the process grouping and optional filters that all Charts in the View use. For more details on Views, refer to Chapter 3.

While the former show trace data in graphical form over a horizontal axis representing runtime, the latter show statistical data. All these Charts are found under **Views Menu**⟶**Charts**. Opening a file in the Intel® Trace Analyzer, displays a View containing the Function Profile Chart (showing the Flat Profile tab) for the opened file.

The following sections describe each Chart in detail. For each Chart there is a subsection about Mouse Hovering, the context menu, the settings dialog box, the effects of filtering and tagging and the effects of aggregation, if it is applicable.

4.1. Event Timeline

The Event Timeline provides a graphical display of the individual process' activities.



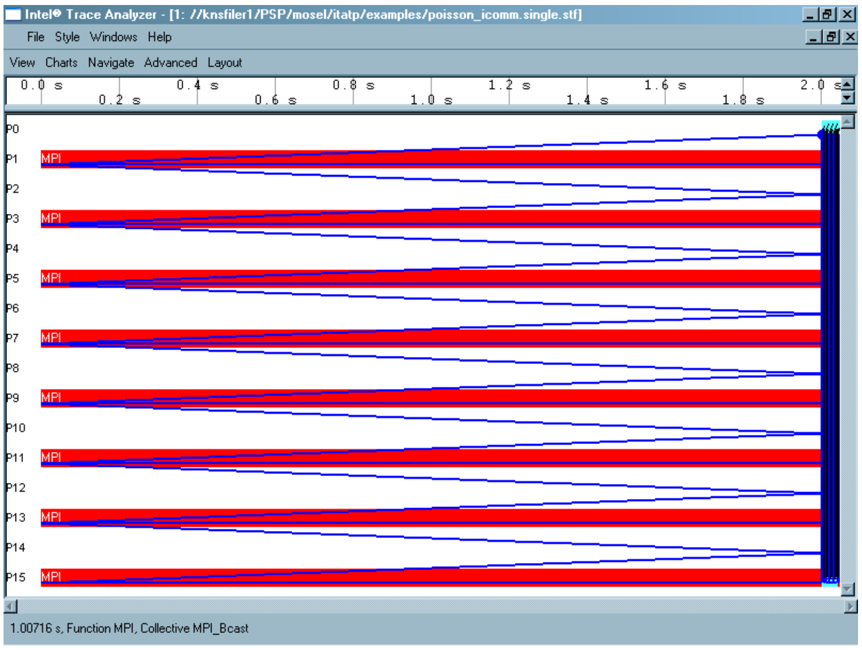
2.01214 s, Message Sender: P13 Receiver: P14

Figure 4-1. The Event Timeline

The horizontal axis represents an interval of the runtime of the inspected program. Vertical bars show the function in which the process is. The bars consist of rectangles labeled with the current function's name and in its color. Black lines indicate messages sent between processes; they connect the sender and receiver processes.Blue lines, forming a grid, represent collective operations, such as broadcast or reduce operations. The status bar at the bottom of this panel shows information on events under the mouse cursor. By default, the entire runtime of the trace is visible. To get a better impression on the visualization primitives, use zooming. Zooming and navigation are explained in detail in Section 3.1.3.

4.1.1. Mouse Hover

When the mouse pointer hovers over the Event Timeline, its exact position in terms of time is shown in the status bar. The status bar also shows which function, collective operations or messages are under the mouse cursor at the moment. Looking at the status bar of the Event Timeline Chart in Figure 4-2, it is seen that the mouse is currently at 1.00716 seconds, in the function *MPI* and in the collective operation *MPI_Bcast*.


Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
0.0 s
0.2 s
0.4 s
0.6 s
0.8 s
1.0 s
1.2 s
1.4 s
1.6 s
1.8 s
2.0 s
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
MPI

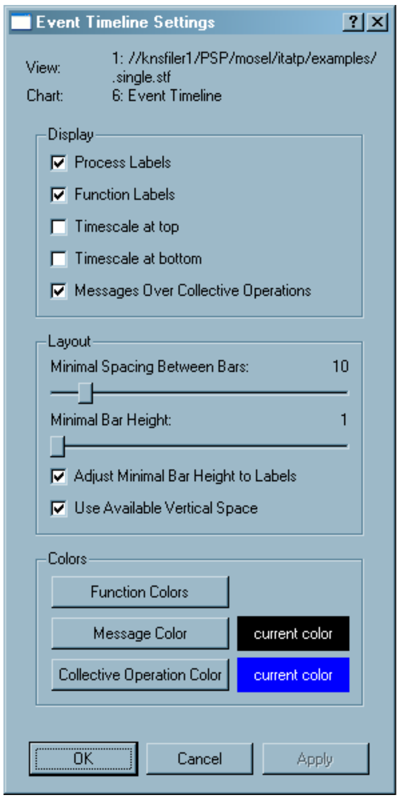
1.00716 s, Function MPI, Collective MPI_Bcast

Figure 4-2. Mouse Hovering in the Event Timeline

4.1.2. Event Timeline Settings

Use the Event Timeline Settings to change the display settings, the layout settings and the colors of the Event Timeline Chart. The display options, which are enabled using check boxes, include

- *Process Labels:* Displays the name of the process or process group to the left.
- *Function Labels:* Displays the name of the function or function group inside the colored bars
- *Timescale at top:* Displays the timescale above the Chart
- *Timescale at bottom:* Displays the timescale below the Chart
- *Messages over Collective Operations:* This specifies what is drawn first. If enabled, messages are drawn over the collective operations such that the messages are seen and the collective operations are concealed. If disabled, the collective operations are drawn over the messages.


Event Timeline Settings

View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.stf
Chart: 6: Event Timeline

Display
- [✓] Process Labels
- [✓] Function Labels
- [ ] Timescale at top
- [ ] Timescale at bottom
- [✓] Messages Over 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

OK Cancel Apply

Figure 4-3. Settings dialog box for the Event Timeline

The Layout option consists of two sliders and two check boxes. The slider *Minimal Spacing Between Bars* adjusts the space between the function bars. The slider *Minimal Bar Height* adjusts the height of the function bar itself. Fot both, the unit is pixels.

The *Adjust Minimal Bar Height to Labels* check box alters the size of the bars. Check this box to make the bars tall enough to display a function label. This option is checked by default.

The check box *Use Available Vertical Space* influences the overall vertical layout of the bars and is also checked by default. For a better understanding, refer to Figure 4-4 where the Event Timeline is shown without checking the given option and Figure 4-5 where the *Use Available Vertical Space* is checked.

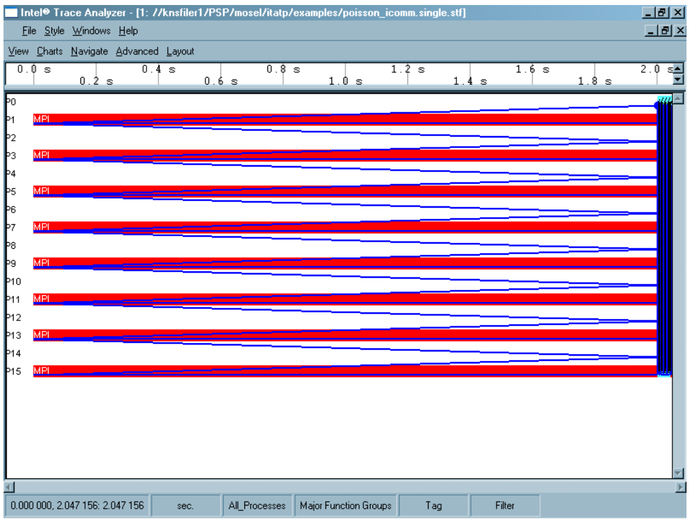

Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
0.0 s 0.2 s 0.4 s 0.6 s 0.8 s 1.0 s 1.2 s 1.4 s 1.6 s 1.8 s 2.0 s
P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15
MPI
0.000 000, 2.047 156: 2.047 156
sec.
All_Processes
Major Function Groups
Tag
Filter

Figure 4-4. Event Timeline: ***Use Available Vertical Space*** **unchecked**

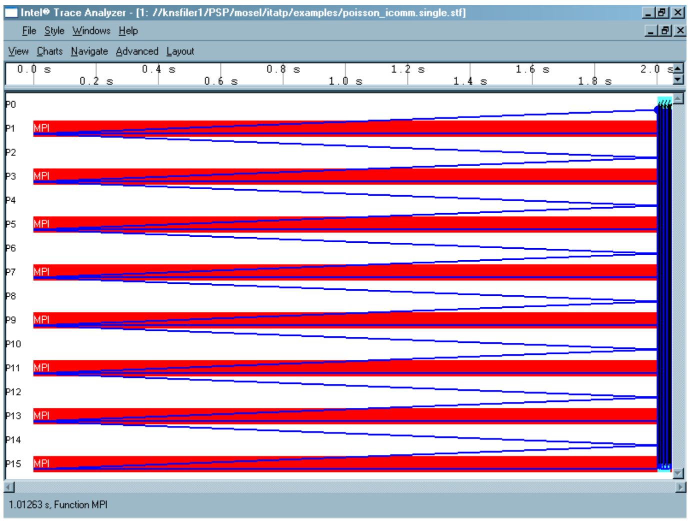

Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
0.0 s 0.2 s 0.4 s 0.6 s 0.8 s 1.0 s 1.2 s 1.4 s 1.6 s 1.8 s 2.0 s
P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15
MPI
1.01263 s, Function MPI

Figure 4-5. Event Timeline: ***Use Available Vertical Space*** **checked**

To change the colors of the functions, messages or collective operations, use the three *Colors* buttons at the bottom of the Settings dialog box. *Function Colors* call up the Function Group Color Editor (Section 5.5). To change the color in which the messages are displayed, click on the *Message Color* button. This opens a dialog box where the required color is chosen. Change the color of the *Collective Operations* in the same way if need be.

Colors chosen for messages and collectives are local to the Event Timeline Chart. Colors chosen for functions or function groups are shared by all Charts and Views belonging to the same trace file.

4.1.3. The Context Menu

Apart from the general options common to all Charts (see Section 4.7), the context menu of the Event Timeline provides details pertaining to the function, the message or the Collective Operation. For example, to access details about a particular message in the Event Timeline, right-click on the message and select the option *Details on Message*. This opens a new dialog box displaying information on sender, receiver and other message attributes. This is further explained in Section 5.6.

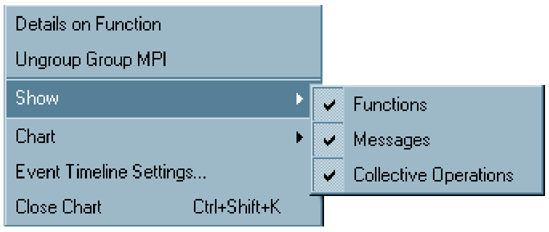


Figure 4-6. Event Timeline: context menu example

Furthermore, the context menu has an entry to ungroup/regroup the function, which works the same way in every Chart. To display functions, messages and/or collective operations, select *Show* from the context menu. Select/deselect one or more of the above from the sub-menu that opens on clicking *Show*.

The context menu entries that are common to all Charts are explained in Section 4.7.

4.1.4. Filtering and Tagging

Tagged functions in the Event Timeline are shown with a frame around them and with the function label rendered in a bold font. Tagged messages and collectives are shown with wider lines.

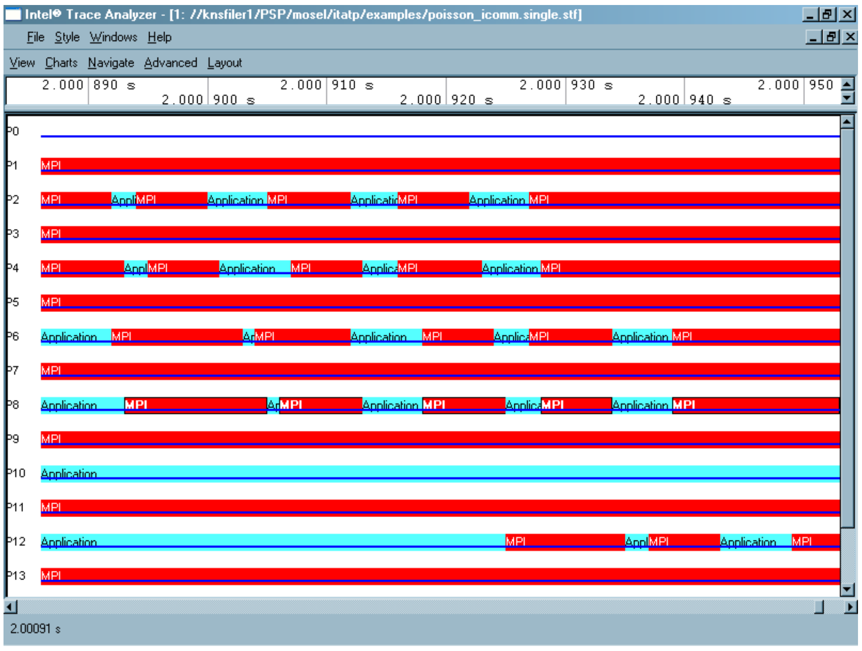


Figure 4-7. Tagging functions in a process in the Event Timeline

Figure 4-7 shows an Event Timeline where the MPI function in process number 8 (P8) is tagged. On filtering the Event Timeline with the same clause (MPI in P8), only the MPI functions of P8 pass the filter and are displayed, while all other functions and processes are filtered out as shown in Figure 4-8. More on tagging and filtering is available under Section 7.3.

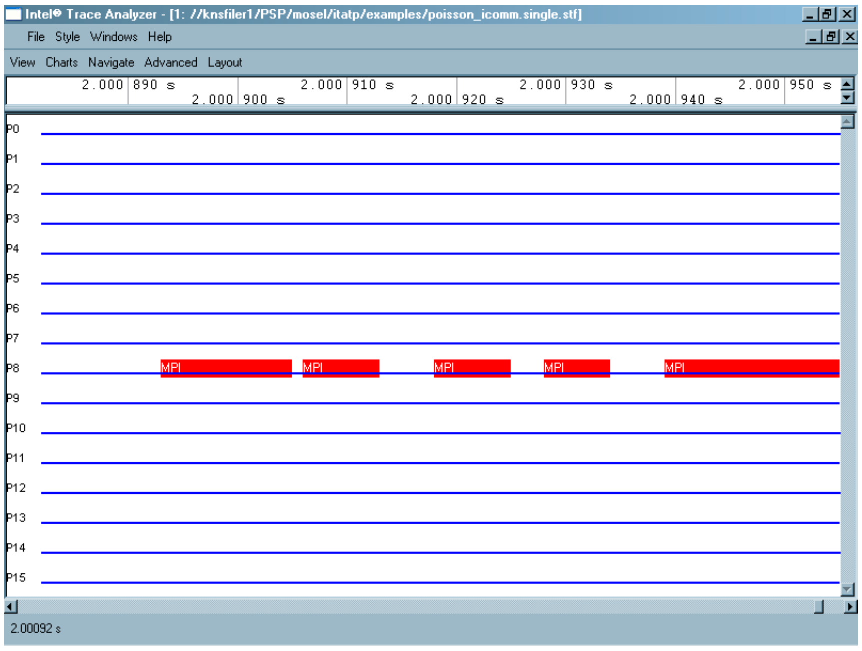


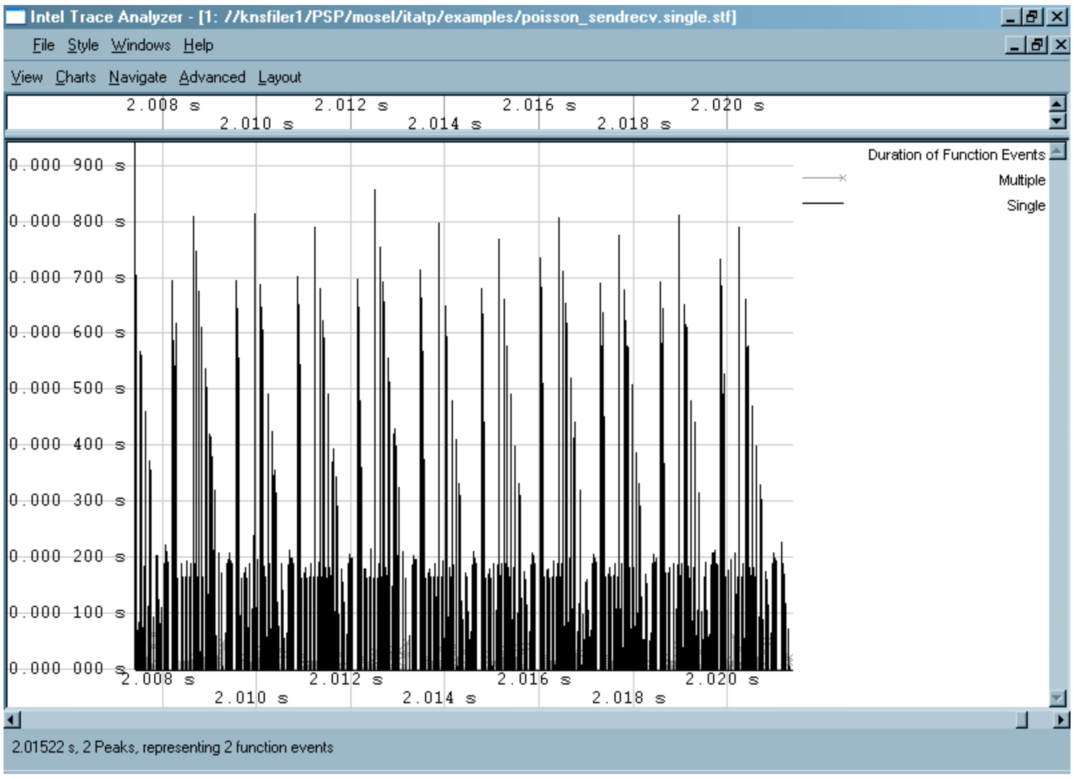
Figure 4-8. Filtering functions in a process in the Event Timeline

4.2. Qualitative Timeline

The Qualitative Timeline (**Views Menu**⟶**Charts**⟶**Qualitative Timeline**) shows event attributes like the data volume of messages as they occur over time. The value of this attribute is plotted along the y-axis while time is plotted along the x-axis. Select the required event type and attributes from the context menu. It helps detect patterns and irregular behavior such as extreme deviations or long-term changes in attribute values.

For the Poisson example, Figure 4-9 shows that the Qualitative Timeline gives a good impression of the pattern of function events **Context Menu**⟶**Events to show**. Zooming to a bunch of iterations again shows the staircase pattern observed above (see Section 1.3). Showing the transfer rates for messages results in a very instructive pattern for the given trace file.

A peak in the Qualitative Timeline either represents a single event (denoted as *Single* in the legend) or several events grouped together (denoted as *Multi* in the legend). Refer to Section 7.1 to find details on the merging of events.

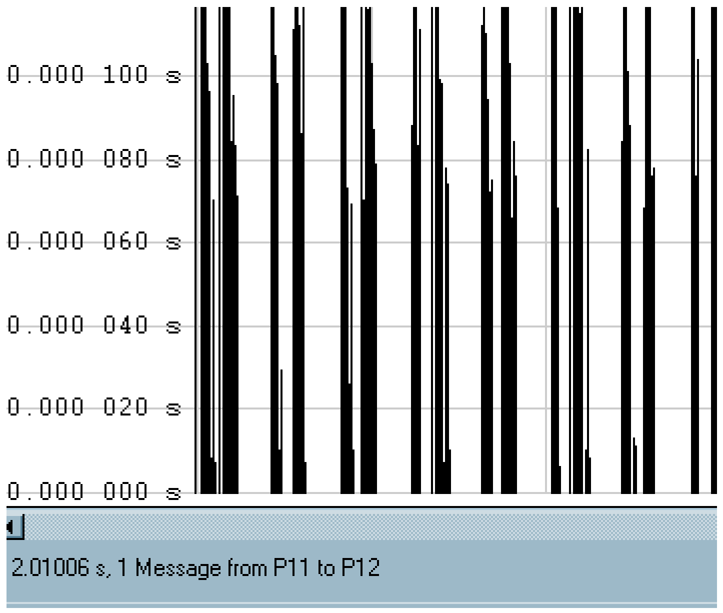


2.01522 s, 2 Peaks, representing 2 function events

Figure 4-9. Qualitative Timeline

4.2.1. Mouse Hover

When the mouse hovers over the Qualitative Timeline Chart, the value of the x-axis (i.e. the time) for the given position is shown in the status bar. The status bar also shows details regarding the given position of the mouse. For example, when the Qualitative Timeline displays the *Message* attribute, the status bar shows how many messages are represented at that point. In case there is only one message, the sender and the receiver of the message are displayed as well. In the case where function events are displayed, mouse hovering indicates which Function/Group is present at the given position.

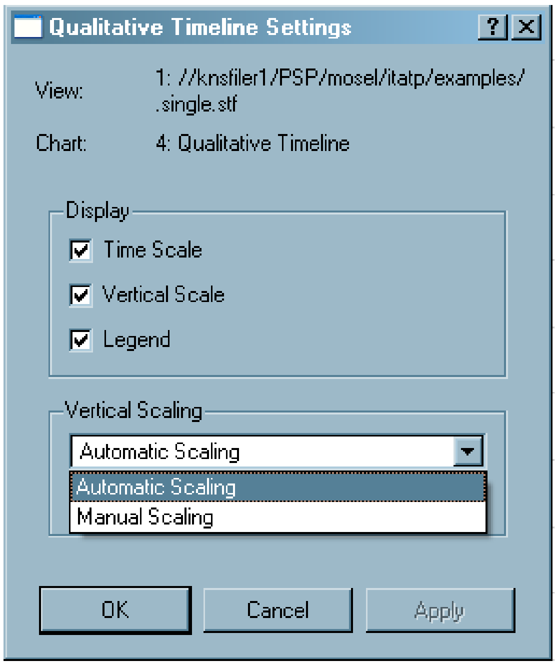


2.01006 s, 1 Message from P11 to P12

Figure 4-10. Mouse Hover

4.2.2. Qualitative Timeline Settings

The settings dialog box of the Qualitative Timeline consists of a *Display* group with check boxes and a *Vertical scaling* combo box. Use the combo box to adjust the vertical scaling of the timeline.



Qualitative Timeline Settings

View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.stf

Chart: 4: Qualitative Timeline

Display

- [x] Time Scale
- [x] Vertical Scale
- [x] Legend

Vertical Scaling

Automatic Scaling

Figure 4-11. The Qualitative Timeline settings dialog box

The Display Group specifies which scales to show and whether an own legend needs to be displayed for the timeline. By default, the vertical scale and the legend are shown.

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

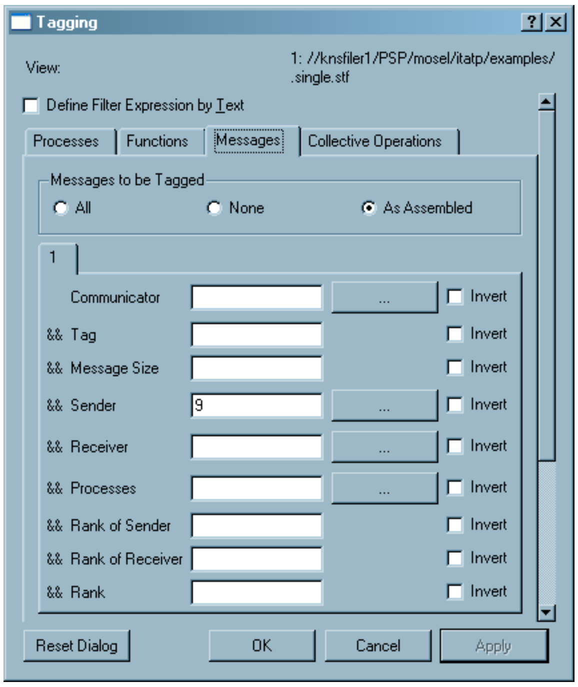
4.2.3. Context Menu

The entry **Context Menu**⟶**Events to show** allows to choose the event type from *Function Events*, *Messages* and *Collective Ops* via a sub-menu.

The entry **Context Menu**⟶**Attribute to show**⟶**xxx** allows to choose the particular attribute value of the event from *Duration*, *Transfer Rate* or *Data Volume* via a sub-menu. Note that not all attributes are available for all event types.

4.2.4. Filtering and Tagging

Tagged items in the Qualitative Timeline are highlighted by coloring them red. For example, to tag all messages sent by P9, open the *Tagging* dialog box (Section 5.2) and go to the Messages tab.


Tagging

View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.stf

☐ Define Filter Expression by Text

Processes | Functions | Messages | Collective Operations

Messages to be Tagged: All None [✓ As Assembled]

1

Communicator		☐ Invert
&& Tag		☐ Invert
&& Message Size		☐ Invert
&& Sender	9	☐ Invert
&& Receiver		☐ Invert
&& Processes		☐ Invert
&& Rank of Sender		☐ Invert
&& Rank of Receiver		☐ Invert
&& Rank		☐ Invert

Figure 4-12. Tagging in the Qualitative Timeline

Here, under the group *Messages to be Tagged* select the radio button titled *Custom*. Specify whatever needs to be tagged in the given field. Figure 4-12 shows that P9 is tagged, resulting in the timeline shown in Figure 4-13.

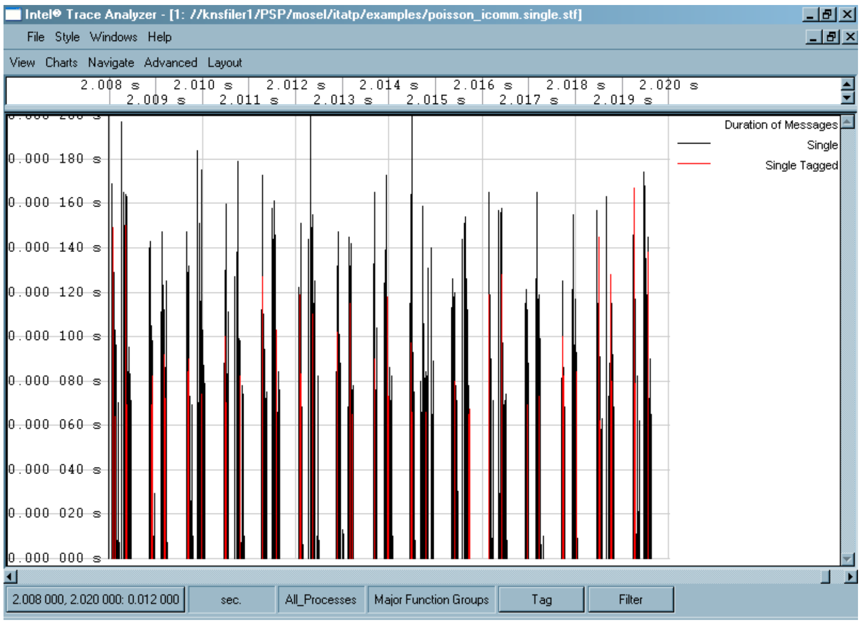


Figure 4-13. Qualitative Timeline with tagged messages

Similarly, to filter out all messages except those sent by P9, use the *Filtering* dialog box (Section 5.1) for details regarding the *Filtering dialog box*. It results in the following timeline:

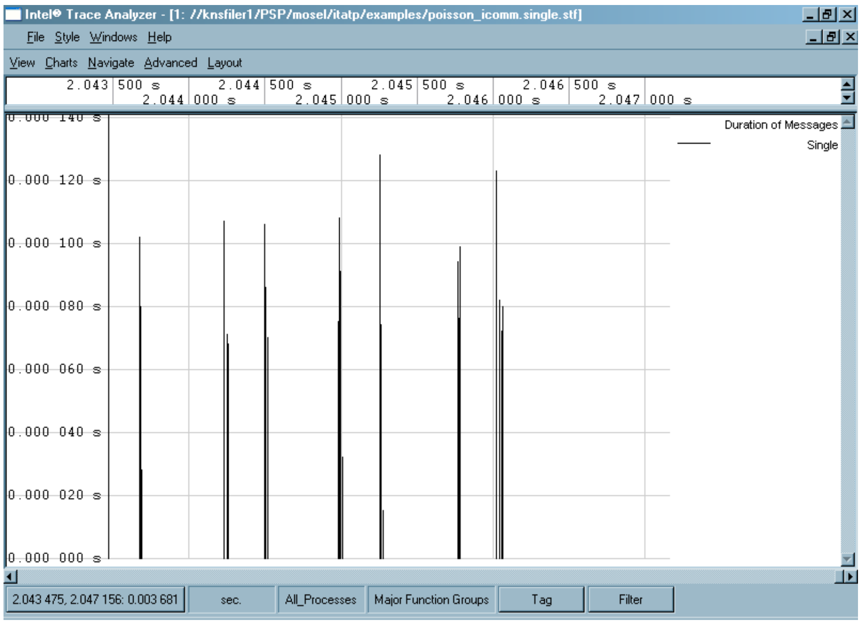


Figure 4-14. Qualitative Timeline after filtering

Using tagging in the Qualitative Timeline is an efficient way to find specific events that occur infrequently. this is because it is guaranteed that a grouped multiple event is tagged if at least one of the singular events it represents matches the tagging filter expression.

4.3. Quantitative Timeline

The Quantitative Timeline (**Views Menu**⟶**Charts**⟶**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. Along the time axis, the 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.

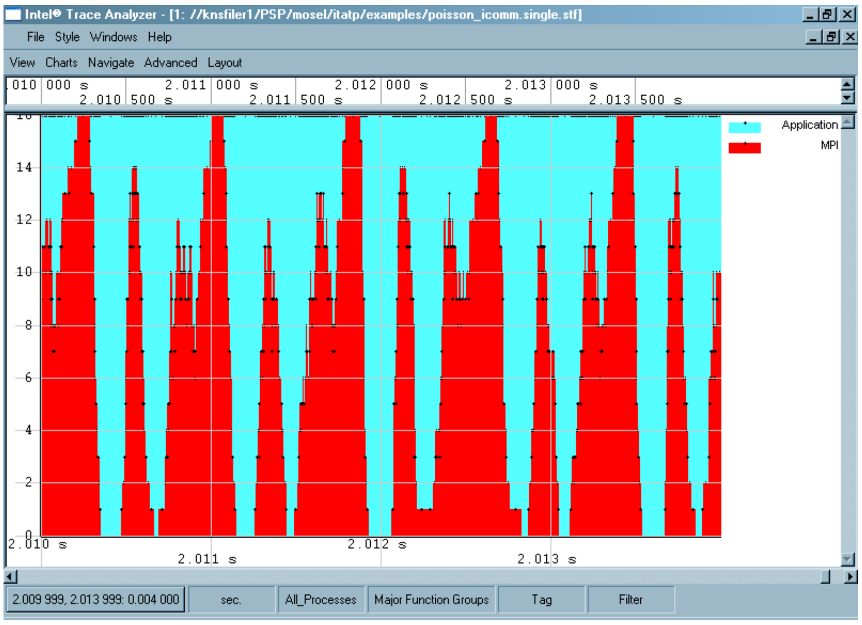


Figure 4-15. Quantitative Timeline

4.3.1. Mouse Hover

When the mouse hovers over the Quantitative Timeline, the status bar displays the present position of the cursor with respect to time. It also shows details pertaining to the function group under the cursor. Figure 4-15 shows that the mouse is currently at 2.012 seconds and that it is over the function group Application that is executed by sixteen processes at this point in time.

4.3.2. Quantitative Timeline Settings

The Settings dialog box in the Quantitative Timeline has two tabs - the *Preferences* tab and the *Functions* tab.

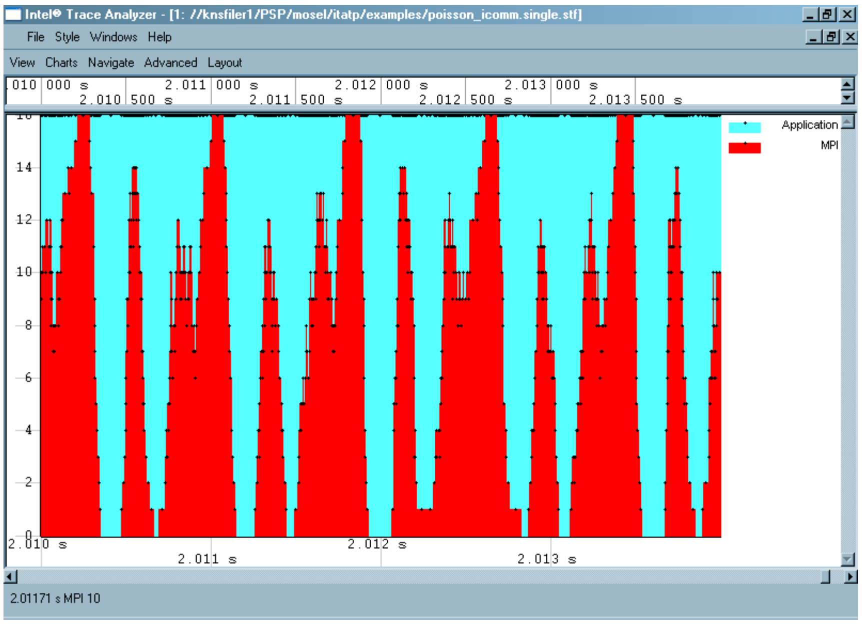

Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
Application
MPI
2.01171 s MPI 10

Figure 4-16. Quantitative Timeline without a grid

- *Preferences Tab*

  Use the options in this tab to adjust the display options and the scaling of the timeline. It has a *Display* group and a *Vertical Scaling* group. Under the *Display* group, there are six check boxes. These are:

  - *Time Scale:* This displays the time scale along the x-axis.
  - *Vertical Scale:* This scale is on the y-axis. By default, this option is enabled so that the scale is shown.
  - *Legend:* This shows the legend in the right margin of the timeline. This is also enabled by default.
  - *Adjust Graphics to Legend Height:* This forces the size of the diagram to be large enough to show all legend items. The checkbox is checked only if the *Legend* option is enabled.
  - *Frames:* Frames give an outline to the bars. The usage of frames becomes visible only when zooming in very closely such that each bar is separated from the other. Figure 4-17 illustrates the Quantitative Timeline with frames. In this figure, the *Application* function group is shown with a black outline, while the *MPIs* are shown with a white outline.
  - *Grid:* Use this checkbox to turn the grid on or off. The grid is drawn on top of the data and is aligned with the ticks on the scales. By default, this checkbox is activated. The difference between a Chart with grid and without grid is seen when comparing Figure 4-15, which shows the grid and Figure 4-16, where the grid is disabled.

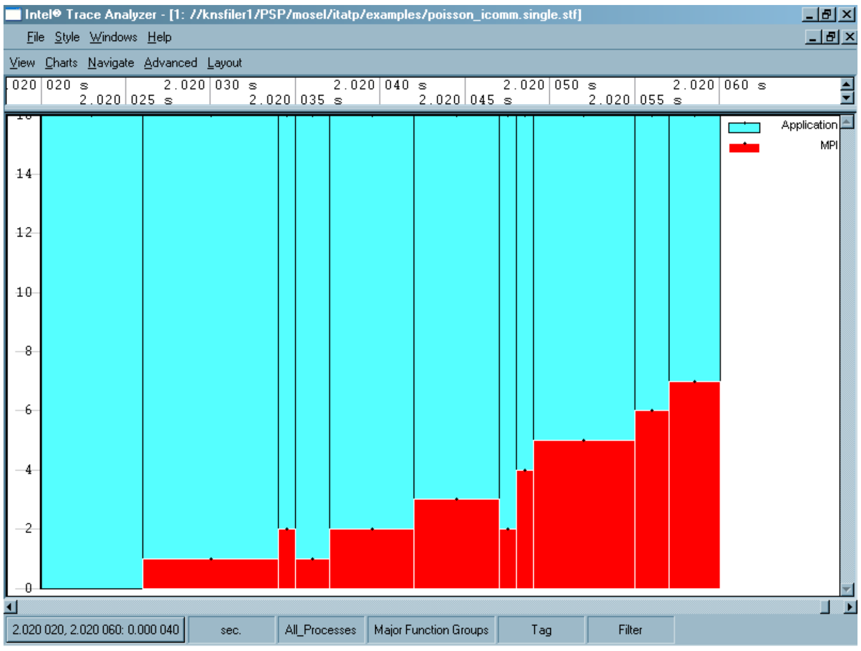
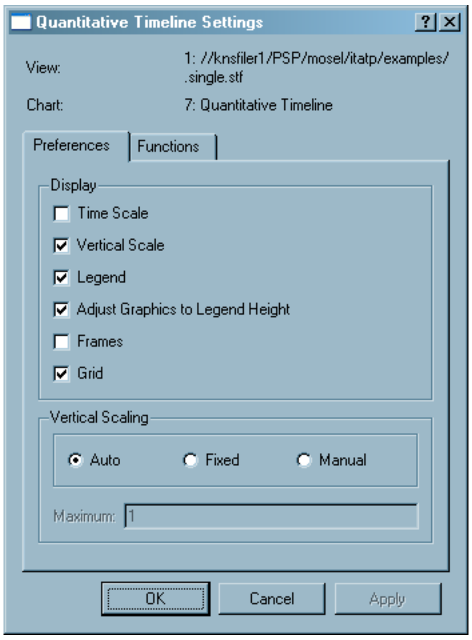


Figure 4-17. Using frames in the Quantitative Timeline

Vertical Scaling

The vertical scaling option functions the same way as in the Qualitative Timeline (see Section 4.2.2.


Quantitative Timeline Settings

View: 1: //knsfiler1/PSP/mosel/itatp/examples/ .single.stf

Chart: 7: Quantitative Timeline

Preferences | Functions

Display
- ☐ Time Scale
- ☑ Vertical Scale
- ☑ Legend
- ☑ Adjust Graphics to Legend Height
- ☐ Frames
- ☑ Grid

Vertical Scaling
[✓ Auto] Fixed Manual

Maximum: 1

OK Cancel Apply

Figure 4-18. The Quantitative Timeline settings dialog box

- *The Functions Tab*

  The *Functions Tab* selects the functions to be hidden/displayed. It is also possible to change the stacking order with this tab.

  Radio buttons in this tab specify whether each individual process is counted (*Count individual Processes*) or whether merely the uppermost entries of *Group All_Processes*(*Count Top Level Entries of Group All_Processes Only*) are counted.

4.3.3. The Context Menu

The context menu entry *Ungroup/Regroup* in the Quantitative Timeline provides the option of ungrouping the given function group, like in the Function Profile (refer to Section 4.4.5.2). Similarly, it regroups the function group.

The *Hide* option in the context menu conceals the chosen activity. To unhide all the hidden items, use the check boxes in the *Functions* tab of the *Settings Dialog* box.

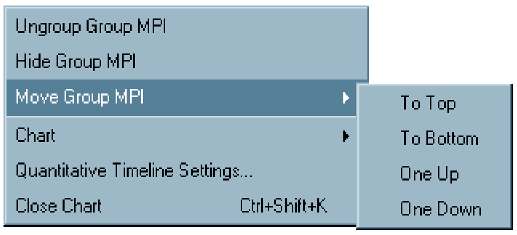


Figure 4-19. Quantitative Timeline: context menu

The *Move Group* entry in the context menu changes the position of the different groups. The opted group can be moved to the top, to the bottom, upward by one position or downward by one position. Click on the legend to obtain a context menu with these options.

The other context menu entries are explained in Section 4.7.

4.3.4. Filtering and Tagging

Tagging in the Quantitative Timeline is indicated by a mesh pattern being placed over the tagged item. Figure 4-20 shows a Quantitative Timeline for the file `poisson_icomm.single.stf` with the tagged function *MPI_Finalize* on the right-hand side of the display.

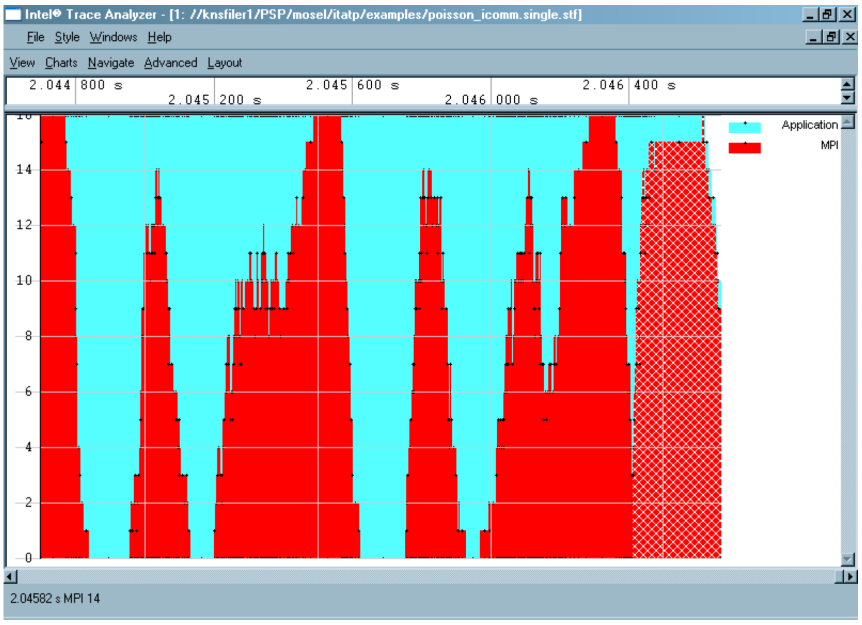

Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
2.044 800 s
2.045 200 s
2.045 600 s
2.046 000 s
2.046 400 s
Application
MPI
14
12
10
8
6
4
2
0
2.04582 s MPI 14

Figure 4-20. Tagging the *MPI_Finalize* function in the Quantitative Timeline

Filtering in the Quantitative Timeline works the same way as in any other Chart. Figure 4-21 shows the result when only *MPI_Finalize* passes the filter.

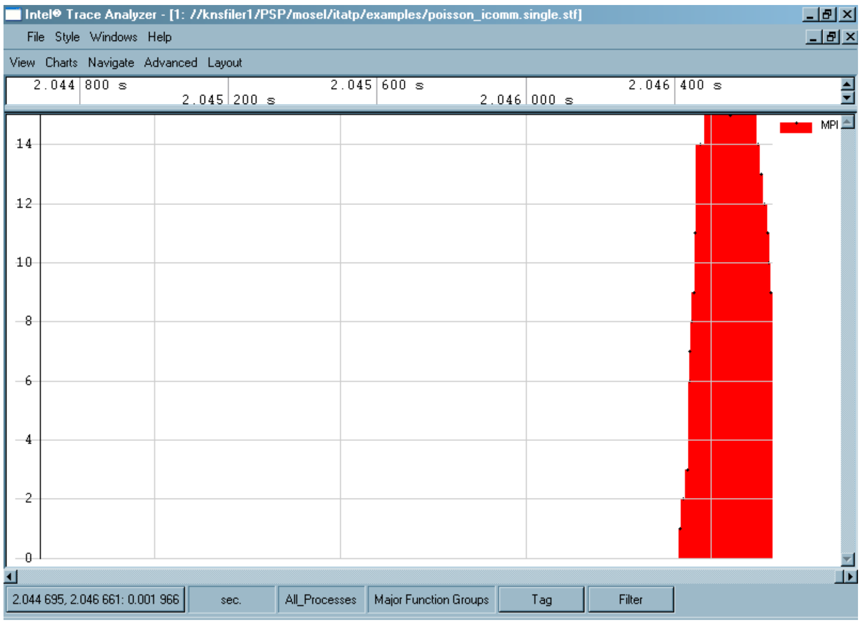
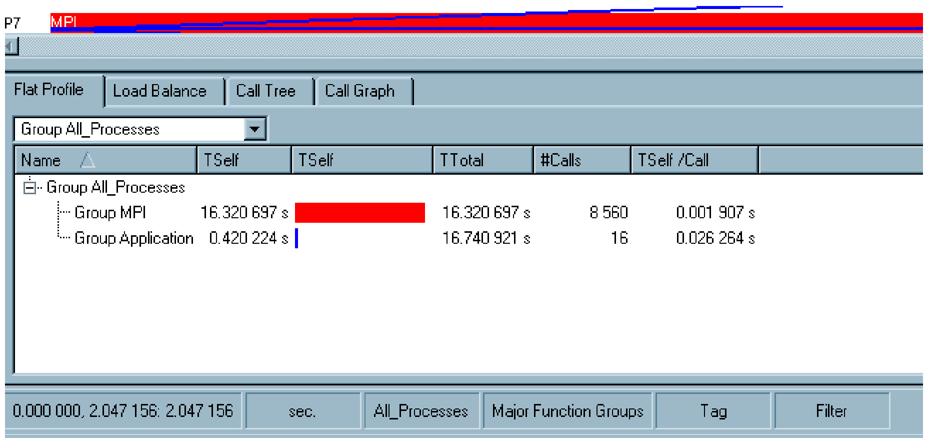


Figure 4-21. Quantitative Timeline after filtering

4.4. The Function Profile Chart

The Function Profile provides detailed profiling information on the performance data. It consists of four different tabs, namely the *Flat Profile* tab, the *Load Balance* tab, the *Call Tree* tab and the *Call Graph* tab.

All four of these tabs use the same column headers with the same semantics, and use 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 column headers refer to Section 4.4.5.1. The order of these columns are adjusted by dragging headers of columns.



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group MPI	16.320 697 s	🟥	16.320 697 s	8 560	0.001 907 s
Group Application	0.420 224 s		16.740 921 s	16	0.026 264 s

Figure 4-22. Function Profile

To sort a list in ascending or descending order, click on a column header. To see which process spends the most time (or the least time) in a function, click *TSelf* and the entries are sorted by this column. The arrow symbol in the column header indicates whether it is arranged in ascending or descending order.

> **Note:** Sorting by the name column does not sort alphabetically. Instead, it sorts in the order given by the layout of the current process or function group.

4.4.1. 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 as defined by the user; in the file `poisson_icomm.single.stf`, this is only *MPI* and *Application*.

The Chart in Figure 4-22 shows that most of the time was spent in MPI, which is considered as pure overhead. To see the distribution of execution time over the individual MPI routines, right-click on the MPI entry and select *Ungroup Group MPI* from the context menu as shown in Figure 4-23.

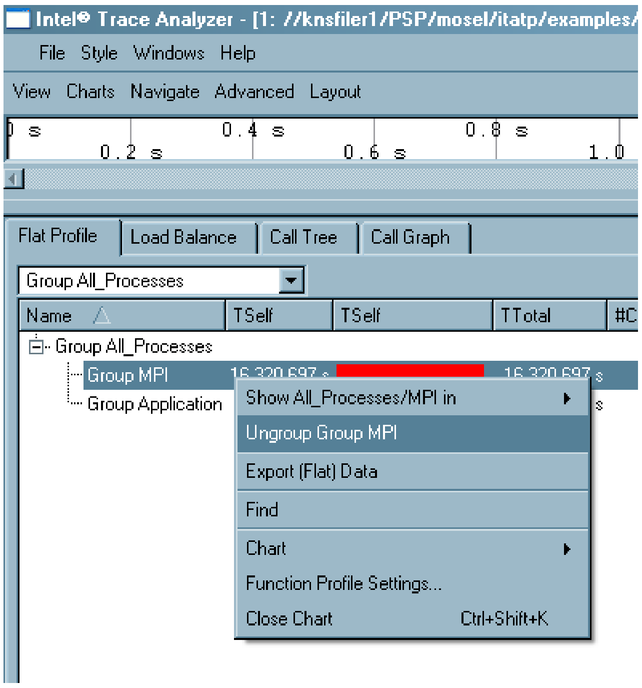
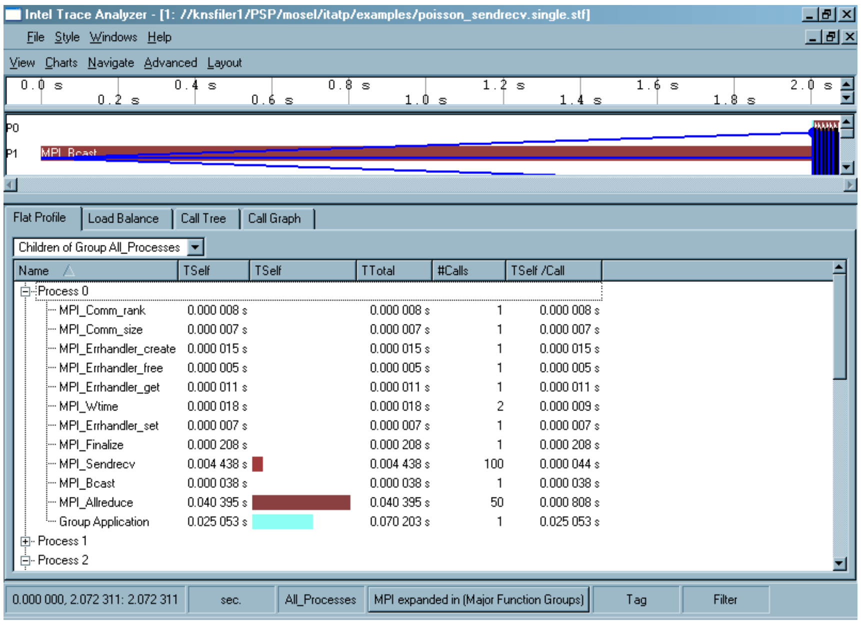


Figure 4-23. Ungrouping the function group MPI via the context menu

This causes the single MPI entry to be replaced by several entries - one for each MPI function (see Figure 4-24). To regroup the children of MPI, right-click on a child and choose *Regroup MPI* from the context menu or select *Major Function Groups* from the *Function Group Editor* ((***Views Menu***$\longrightarrow$***Advanced***$\longrightarrow$***Function Aggregation***).


Intel Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_sendrecv.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
P0
P1
MPI_Bcast
Flat Profile
Load Balance
Call Tree
Call Graph
Children of Group All_Processes

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Process 0					
MPI_Comm_rank	0.000 008 s		0.000 008 s	1	0.000 008 s
MPI_Comm_size	0.000 007 s		0.000 007 s	1	0.000 007 s
MPI_Errhandler_create	0.000 015 s		0.000 015 s	1	0.000 015 s
MPI_Errhandler_free	0.000 005 s		0.000 005 s	1	0.000 005 s
MPI_Errhandler_get	0.000 011 s		0.000 011 s	1	0.000 011 s
MPI_Wtime	0.000 018 s		0.000 018 s	2	0.000 009 s
MPI_Errhandler_set	0.000 007 s		0.000 007 s	1	0.000 007 s
MPI_Finalize	0.000 208 s		0.000 208 s	1	0.000 208 s
MPI_Sendrecv	0.004 438 s	🟥	0.004 438 s	100	0.000 044 s
MPI_Bcast	0.000 038 s		0.000 038 s	1	0.000 038 s
MPI_Allreduce	0.040 395 s	🟥	0.040 395 s	50	0.000 808 s
Group Application	0.025 053 s	🟦	0.070 203 s	1	0.025 053 s
Process 1					
Process 2					

0.000 000, 2.072 311: 2.072 311 | sec. | All_Processes | MPI expanded in (Major Function Groups) | Tag | Filter

Figure 4-24. Flat Profile after ungrouping MPI

The default settings ensure that all statistics are summarized into a single profile. Each of these Charts also provide the option of viewing the data for each process. To do this, use the combo box at the top of the tab as shown in Figure 4-25. For example, selecting *Children of Group All_Processes* results in Figure 4-26. The processes are now listed as the top-level entries in the tree (first column). To expand and collapse the processes of interest, use the plus and minus handles (see Figure 4-26).

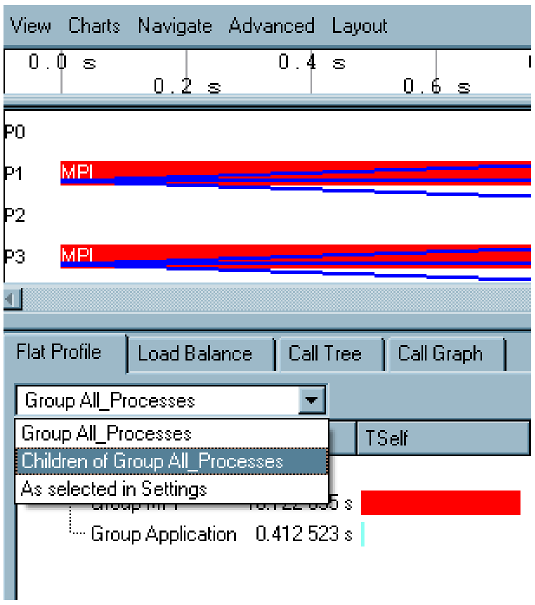
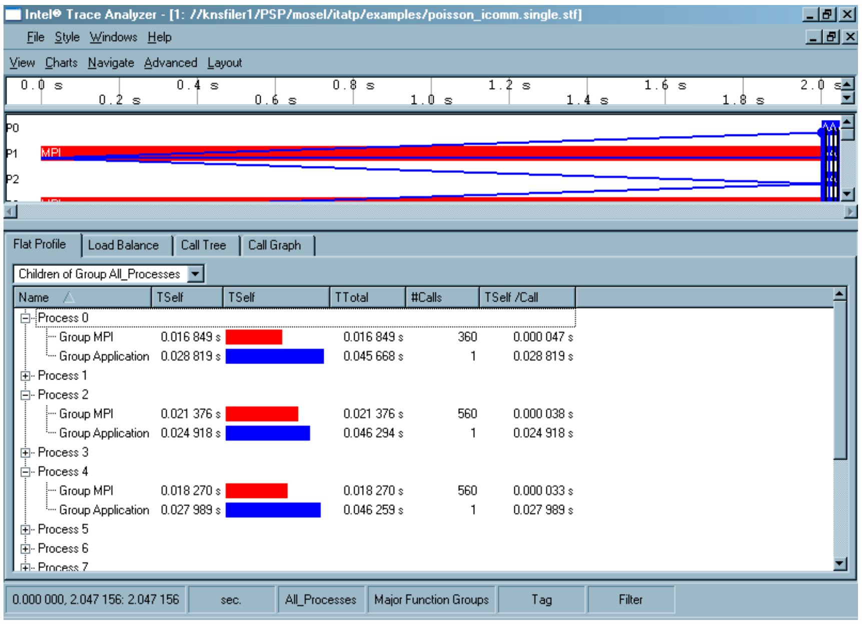


Figure 4-25. Selecting Profiles per process


Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
Flat Profile
Load Balance
Call Tree
Call Graph
Children of Group All_Processes

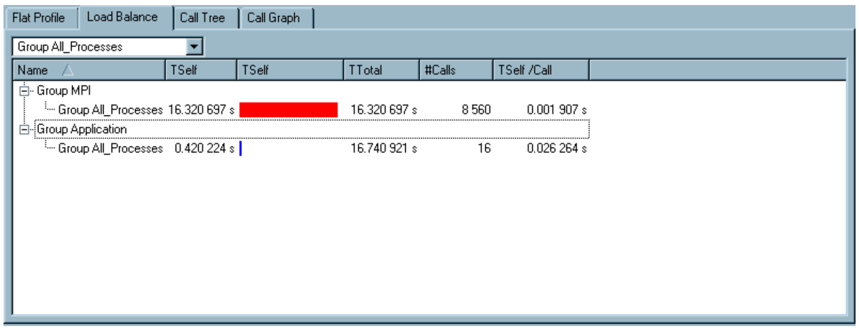
Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Process 0					
Group MPI	0.016 849 s	🟥	0.016 849 s	360	0.000 047 s
Group Application	0.028 819 s	🟦	0.045 668 s	1	0.028 819 s
Process 1					
Process 2					
Group MPI	0.021 376 s	🟥	0.021 376 s	560	0.000 038 s
Group Application	0.024 918 s	🟦	0.046 294 s	1	0.024 918 s
Process 3					
Process 4					
Group MPI	0.018 270 s	🟥	0.018 270 s	560	0.000 033 s
Group Application	0.027 989 s	🟦	0.046 259 s	1	0.027 989 s
Process 5					
Process 6					
Process 7					

0.000 000, 2.047 156: 2.047 156 | sec. | All_Processes | Major Function Groups | Tag | Filter

Figure 4-26. Showing children of process group All Processes

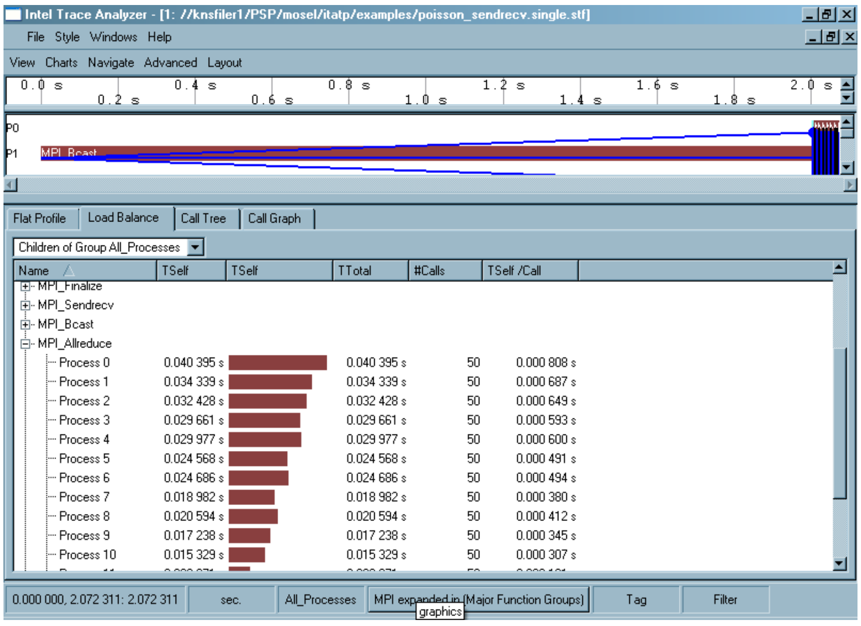
4.4.2. Load Balance

The *Load Balance* tab displays the same data as the *Flat Profile* except that it is grouped by function instead of by process. The *Load Balance* tab compares the profiles of the same function across several processes. Here, the top level entries of the tree, given in the first column, are functions. Figure 4-28 shows that *TSelf* for *MPI_Allreduce* is pretty balanced across processes.



Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group MPI					
Group All_Processes	16.320 697 s		16.320 697 s	8 560	0.001 907 s
Group Application					
Group All_Processes	0.420 224 s		16.740 921 s	16	0.026 264 s

Figure 4-27. The Load Balance tab


Intel Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_sendrecv.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
0.0 s
0.2 s
0.4 s
0.6 s
0.8 s
1.0 s
1.2 s
1.4 s
1.6 s
1.8 s
2.0 s
P0
P1
MPI_Bcast
Flat Profile
Load Balance
Call Tree
Call Graph
Children of Group All_Processes

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
MPI_Finalize					
MPI_Sendrecv					
MPI_Bcast					
MPI_Allreduce					
Process 0	0.040 395 s		0.040 395 s	50	0.000 808 s
Process 1	0.034 339 s		0.034 339 s	50	0.000 687 s
Process 2	0.032 428 s		0.032 428 s	50	0.000 649 s
Process 3	0.029 661 s		0.029 661 s	50	0.000 593 s
Process 4	0.029 977 s		0.029 977 s	50	0.000 600 s
Process 5	0.024 568 s		0.024 568 s	50	0.000 491 s
Process 6	0.024 686 s		0.024 686 s	50	0.000 494 s
Process 7	0.018 982 s		0.018 982 s	50	0.000 380 s
Process 8	0.020 594 s		0.020 594 s	50	0.000 412 s
Process 9	0.017 238 s		0.017 238 s	50	0.000 345 s
Process 10	0.015 329 s		0.015 329 s	50	0.000 307 s

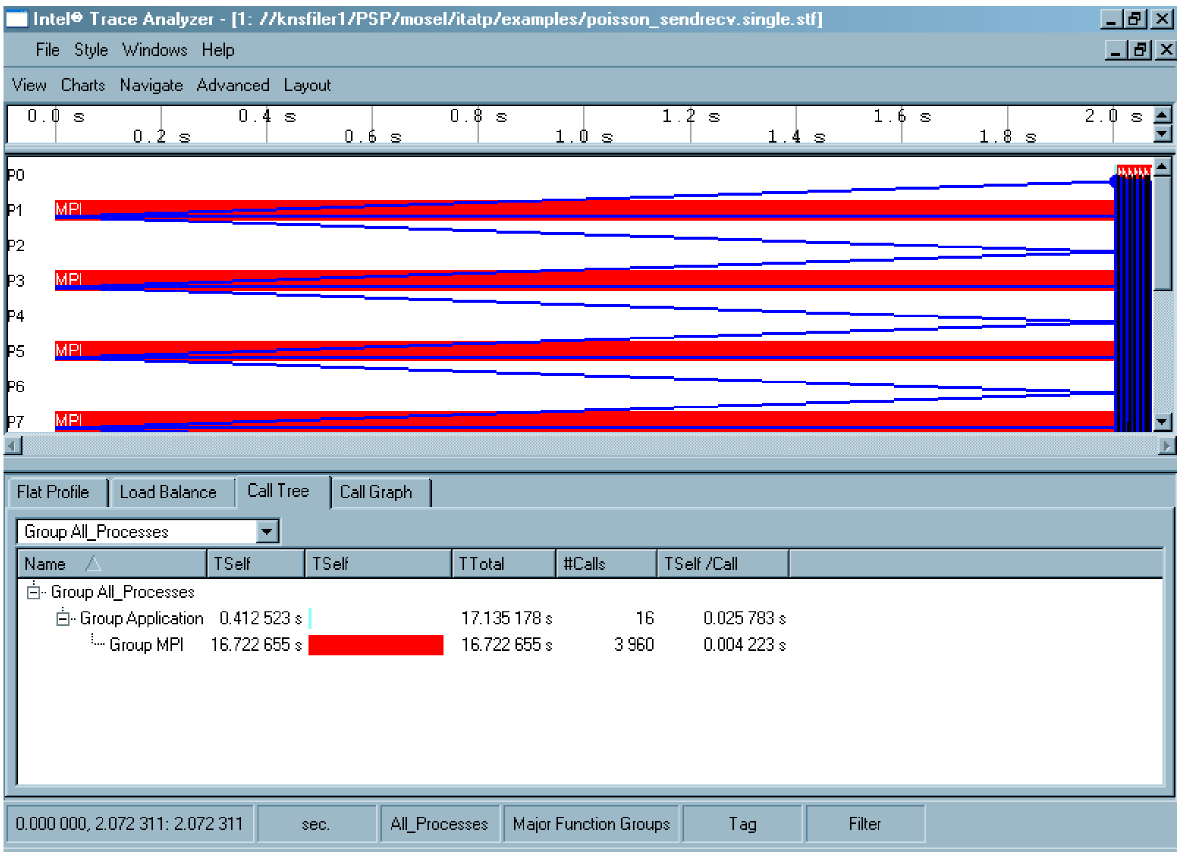
Figure 4-28. Load Balance for MPI_Allreduce

As in the Flat Profile, the Load Balance summarizes the statistics into a single profile using *Group All_Processes*. To view the data for each individual process under a given function, use *Children of Group All_Processes*. Similarly, the functions in the Load Balance tab are ungrouped using *Ungroup/Regroup* on the context menu. Ungrouping displays all major function groups. To group all processes together and to view it as a single profile, select *Group All_Processes*.

4.4.3. Call Tree

When the Load Balance and Flat Profile tabs do not show enough detail, use the Call Tree tab to include calling dependencies in your analysis. The Call Tree tab shows the same information as the Flat Profile and Load Balance, but also includes the calling hierarchy.

Select a certain entry in the Call Tree to focus on it. The focus remains on this entry even when the time interval is changed due to scrolling or zooming. It stays selected and visible when possible. If a corresponding entry is absent for the new time interval, then its parent is selected. This feature is very useful in large and deeply nested call trees.

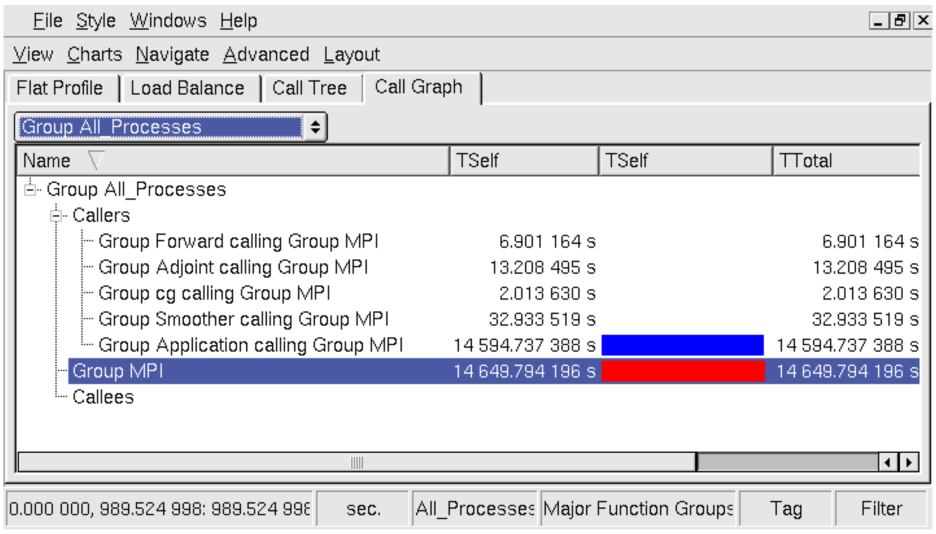


Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group Application	0.412 523 s		17.135 178 s	16	0.025 783 s
Group MPI	16.722 655 s	🟥	16.722 655 s	3 960	0.004 223 s

Figure 4-29. The Call Tree tab

4.4.4. Call Graph

The Call Graph tab only 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*.



Name	TSelf	TSelf	TTotal
Group All_Processes			
Callers			
Group Forward calling Group MPI	6.901 164 s		6.901 164 s
Group Adjoint calling Group MPI	13.208 495 s		13.208 495 s
Group cg calling Group MPI	2.013 630 s		2.013 630 s
Group Smoother calling Group MPI	32.933 519 s		32.933 519 s
Group Application calling Group MPI	14 594.737 388 s	🟦	14 594.737 388 s
Group MPI	14 649.794 196 s	🟥	14 649.794 196 s
Callees			

Figure 4-30. The Call Graph tab

To navigate through the Call Graph, double-click on a caller or callee and press the space bar or Enter key to make the respective function 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 times shown for the callers represent the *time spent in the central function* when called from the respective function.

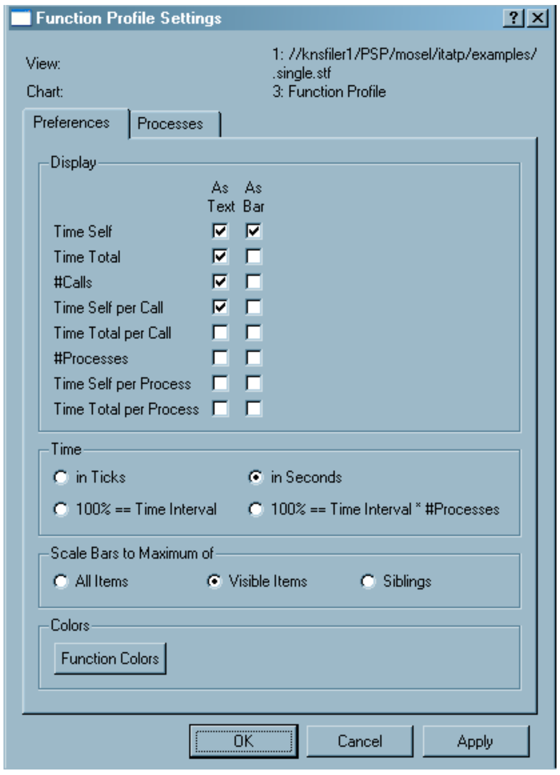
If a function changes the flow control based upon its arguments (by calling different algorithms for example), then it is observed which algorithm is more expensive. In Figure 4-30 it is seen which caller is responsible for the time spent in MPI: it is the function group Application (and not Forward, Adjoin, cg or Smoother). Using the Call Graph this way helps finding places in the code that cause expensive calls, even when the call tree gets too big to navigate through it.

4.4.5. Using the Function Profile

The following sections describe the columns headers of the Function Profile and how to define these headers using the *Function Profile Settings* dialog box.

4.4.5.1. Function Profile Settings

The Function Profile Settings dialog box enables customizing display options for all four tabs of the Function Profile Chart. To access the Settings dialog box, right click and select *Function Profile settings* from the context menu.



Function Profile Settings

View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.stf
Chart: 3: Function Profile

Preferences | Processes

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

Function Colors

OK Cancel Apply

Figure 4-31. The Function Profile settings dialog

- *Preferences Tab*

  Under the *Preferences* tab, there are four groups of options. The first one is the *Display* Group, which consists of check boxes. Use these check boxes to select the attributes to be displayed. There are a total of eight attributes available, out of which four are selected by default (*Time Self*, *Time Total*, *#Calls* and *Time Self per Call*. All eight attributes are described below:

- *TimeSelf (TSelf):* Time spent in the given function, excluding time spent in functions called from it
- *TimeTotal (TTotal):* Time spent in the given function, including time spent in functions called from it
- *#Calls:* Number of calls to this function. This is zero if the calls to the respective function occur outside the current time interval.
- *TimeSelf per Call (TSelf/Call): Timeself* averaged over *#Calls*
- *TimeTotal per Call: TimeTotal* averaged over *#Calls*
- *#Processes:* Number of processes in this function
- *Time Self per Process:* Average time exclusively spent in this function, averaged over *#Processes*
- *Time Total per Process:* Average time spent in this function, averaged over *#Processes*

Using the given check boxes, all of the above attributes are displayed in either text form (the actual numbers) or as a bar graph or both.

Use the radio buttons to specify the format for time (seconds or ticks) or to specify time as a percentage of the time interval.

There are three scaling modes in the Preferences tab and these are given as radio buttons. 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 if they are expanded or not. *Siblings* uses only the maximum of the direct siblings. In all three scaling modes only values from the same column are taken into account.

At the bottom of the *Preferences* tab, there is a *Function Colors* command button. Clicking on this opens the *Function Group Color Editor*.

- *The Processes Tab*

  In the *Processes* tab, select the processes to be displayed in the Chart by enabling the check box of the process. After selecting these in the *Processes* tab, the selected processes are shown in any of the Function Profile tabs by choosing the *As selected in Settings* option from the combo box (See Figure 4-25). An easy way to select all but one process is to choose the process not required and then using the *Invert All* to reverse the selection. Doing this has no influence on the current process group of the View, it only allows to focus the Function Profile on a subset of all processes.

4.4.5.2. The Context Menu

The context menu, obtained by right-clicking on an item, contains a set of operations that are performed on the clicked item and on the Chart as a whole. The context menu adjusts itself to suit the selected entry in the Chart.

The *Show All_Processes/xxx in* entry in the context menu shows the given profile in a different tab. Here, *xxx* stands for the Function group name. For the given example, this would be either the function groups *MPI, Application* or the function group *Other*.

Another context menu entry is the *Ungroup* option. This ungroups the selected group and shows the distribution of execution time over the individual routines, as is illustrated in Figure 4-23 and Figure 4-24. To revert the ungrouping, right-click a child of a recently ungrouped function group and select *Regroup* from the context menu. To restore the summarized display after ungrouping a number of times, it is easier to open the *Function Group Editor* using **Views Menu**⟶**Advanced**⟶**Function Aggregation** and select *Major Function Groups*.

The *Find* entry searches for a process/function (See Section 5.11).

To save the flat profile date in text form, choose the context menu entry *Export (Flat) Data*. This opens a *File Save* dialog box. Specify the filename or choose the file in which to store the data here. This includes all data of the flat profile also taking into account the child processes. The default option is to save it as a .txt file.

Context Menu⟶**Charts** opens another sub-menu, which contains entries to print, save, clone and move the Chart (see Section 4.7).

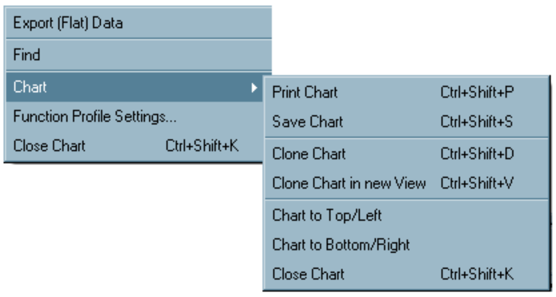
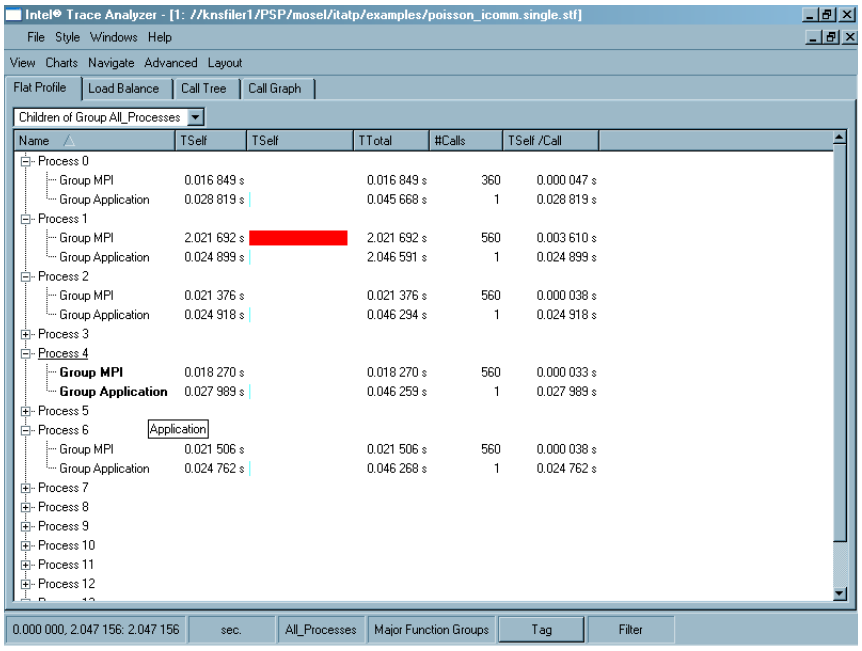


Figure 4-32. A context menu with a submenu

4.4.5.3. Filtering and Tagging

Tagged entries are shown using a bold font for the name column. 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 Section 7.3.


Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
Flat Profile
Load Balance
Call Tree
Call Graph
Children of Group All_Processes
Application
0.000 000, 2.047 156: 2.047 156
sec.
All_Processes
Major Function Groups
Tag
Filter

Name	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Process 0					
Group MPI	0.016 849 s		0.016 849 s	360	0.000 047 s
Group Application	0.028 819 s		0.045 668 s	1	0.028 819 s
Process 1					
Group MPI	2.021 692 s	🟥	2.021 692 s	560	0.003 610 s
Group Application	0.024 899 s		2.046 591 s	1	0.024 899 s
Process 2					
Group MPI	0.021 376 s		0.021 376 s	560	0.000 038 s
Group Application	0.024 918 s		0.046 294 s	1	0.024 918 s
Process 3					
Process 4					
Group MPI	0.018 270 s		0.018 270 s	560	0.000 033 s
Group Application	0.027 989 s		0.046 259 s	1	0.027 989 s
Process 5					
Process 6					
Group MPI	0.021 506 s		0.021 506 s	560	0.000 038 s
Group Application	0.024 762 s		0.046 268 s	1	0.024 762 s
Process 7					
Process 8					
Process 9					
Process 10					
Process 11					
Process 12					

Figure 4-33. Tagged entries in the function profile

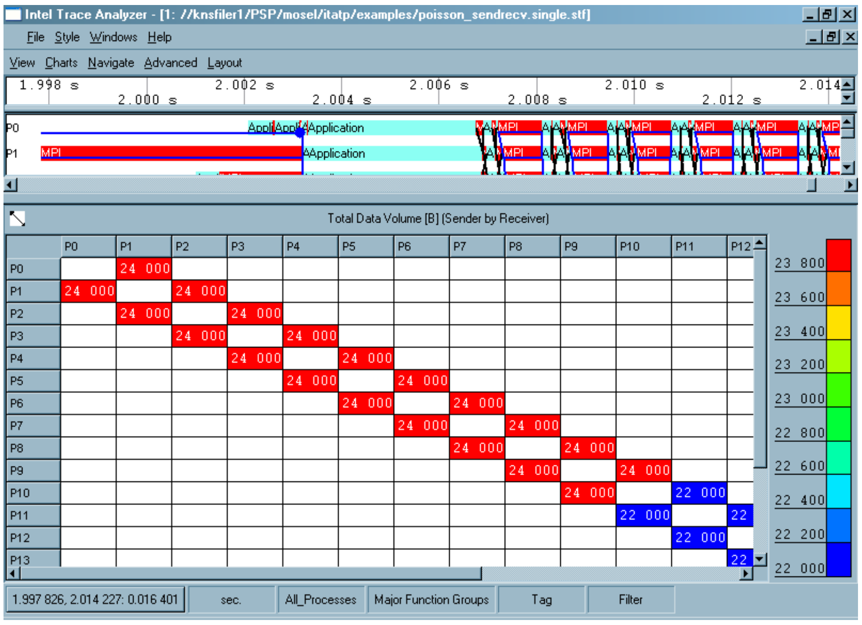
4.5. The Message Profile

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.

The grouping that defines the row and column headers of this matrix and therefore the categorization of the data are changed in the context menu and the settings dialog box. Available groupings in addition to *Sender* and *Receiver* are for example *Tag* and *Communicator*.



Total Data Volume [B] (Sender by Receiver)

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
P0		24 000											
P1	24 000		24 000										
P2		24 000		24 000									
P3			24 000		24 000								
P4				24 000		24 000							
P5					24 000		24 000						
P6						24 000		24 000					
P7							24 000		24 000				
P8								24 000		24 000			
P9									24 000		24 000		
P10										24 000		22 000	
P11											22 000		22
P12												22 000	
P13													22

23 800	🟥
23 600	🟧
23 400	🟨
23 200	🟩
23 000	🟩
22 800	🟩
22 600	🟩
22 400	🟦
22 200	🟦
22 000	🟦

Figure 4-34. The Message Profile

The attribute shown in the cells are chosen via the context menu or the settings dialog box. Apart from the Total Time shown by default there are other time values, transfer rates, volumes, and counts.

You can change the size of the cell by using the slider above the matrix. If the cells are too small to display numeric data, hover your mouse over a square and view data in the status bar.

Click a row or column header or cells to select a rectangular area of the matrix. To zoom, right-click and select a zoom scale from the context menu. To change the position of the row and column headers, hold down the Ctrl key and drag the header to the required position.

4.5.1. Mouse Hover

When the mouse is positioned over any point in the matrix then detailed information for the current position is shown in the View's status bar in the form "$AttributeValue ($RowLabel, $ColumnLabel)". This allows getting exact attribute values even if the cells are configured to be very small or to show no alphanumerical entries at all.

4.5.2. Message Profile Settings

The Settings dialog box provides three tabs: Preferences, Colors and Data.

- *Preferences Tab*

  The *Display* group provides check boxes and radio buttons to configure some visual details. The check boxes *Row Labels* and *Column Labels* switch the respective row and column headers. The check box *Scale* switches the colored scale next to the matrix. The check box *Grid* shows/removes the black grid shown between cells.

  The checkbox *Keep Empty Rows/Columns when using Sender/Receiver Groupings* switches a special feature on/off. This feature is only relevant for the Groupings *Sender* and *Receiver*. For these groupings, a checked state indicates that all processes should always be shown, like for example, showing even empty rows and columns. That keeps the form of the matrix constant and makes it easy to look for patterns in the data. An unchecked state means that 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.

  The radio buttons *Communicator Names* and *Communicator Ids* allow to either see helpful communicator names (if available in the trace file) that may take a lot of valuable screen space or to restrict the display to show only concise communicator ids.

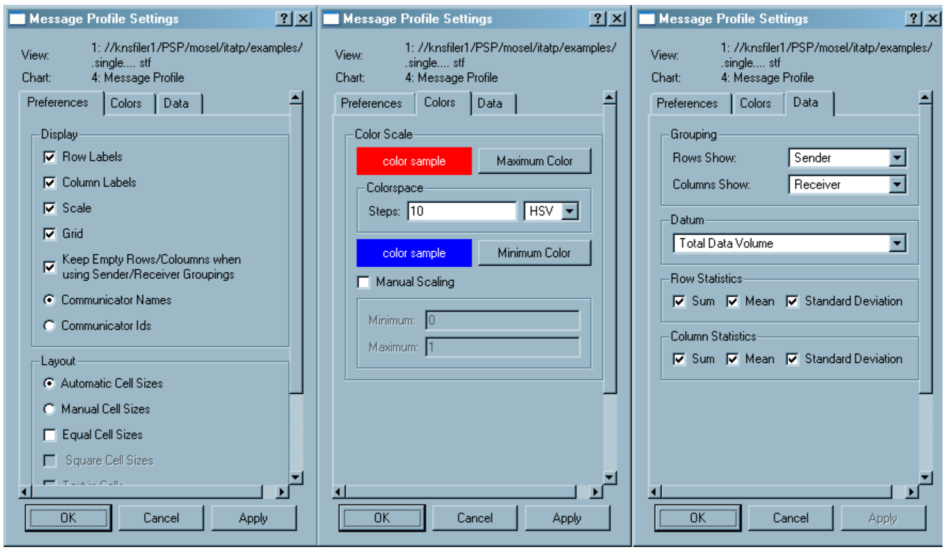
  The *Layout* group allows switching between two fundamentally different modes for the layout of the matrix. By default the mode is *Automatic Cell Sizes* and the cell sizes are adjusted to make all text readable. In this mode checking *Equal Cell Sizes* basically results in equal column widths and enables the check box *Square Cell Sizes* to get square cells. The other options of the *Layout group* are disabled by default.

  Choosing *Manual Cell Sizes* allows specifying the size of the cells in pixels either in the *Cell Size* group at the bottom of the tab or using 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 or if it is switched off entirely by un-checking the check box *Text in Cells*. By default, *Manual Cell Sizes* is checked.

- *Colors Tab*

  The push buttons *Maximum Color* and *Minimum Color* allow choosing the colors for the maximum and minimum attribute values. The text input field allows specifying the number of color steps (1-255).

  The chosen colors are considered as points in a color space and the colors of the scale are interpolated on a line through color space connecting these two points. The combo box to the right of the text input field allows using either the HSV or the RGB color space. HSV is more fancy 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. Choosing white for the minimum and black for the maximum is not at all bad.


Message Profile Settings

View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.... stf
Chart: 4: Message Profile

Preferences | Colors | Data

Display
- [✓] Row Labels
- [✓] Column Labels
- [✓] Scale
- [✓] Grid
- [✓] Keep Empty Rows/Coloumns when using Sender/Receiver Groupings
- (●) Communicator Names
- ( ) Communicator Ids

Layout
- (●) Automatic Cell Sizes
- ( ) Manual Cell Sizes
- [ ] Equal Cell Sizes
- [ ] Square Cell Sizes

Color Scale
- 🟥 color sample — Maximum Color
- Colorspace: Steps: 10, HSV
- 🟦 color sample — Minimum Color
- [ ] Manual Scaling
- Minimum: 0
- Maximum: 1

Grouping
- Rows Show: Sender
- Columns Show: Receiver

Datum
- Total Data Volume

Row Statistics
- [✓] Sum [✓] Mean [✓] Standard Deviation

Column Statistics
- [✓] Sum [✓] Mean [✓] Standard Deviation

OK | Cancel | Apply

Figure 4-35. The three tabs of the settings dialog box

By checking the box *Manual Scaling* it is possible to specify the minimum and maximum values for the color scale in the two text input fields below. This is very convenient when comparing two Message Profile Charts that may live in different Views.

- *Data Tab*

  The *Grouping* group provides two combo boxes to choose the row and column headers or better said to choose how the data is grouped into categories. The groupings for rows and columns are chosen independently. However not all combinations are possible. It is not possible to have the same grouping for rows and columns and it is not possible to have Sender/Receiver at one axis and any one of Sender or Receiver on the other axis.

  The available groupings are:

  - *Sender*

    Categorizes the messages by Sender. The exact labels are defined by the current thread group that is given by the View (see Chapter 3).

  - *Receiver*

    Categorizes the messages by Receiver. The exact labels are defined by the current process group that is given by the View (see Chapter 3).

  - *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 (see Chapter 3).

  - *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. This is seen in Figure 4-36, where only messages with a volume of 2000 bytes were sent.

- *Sending Function*

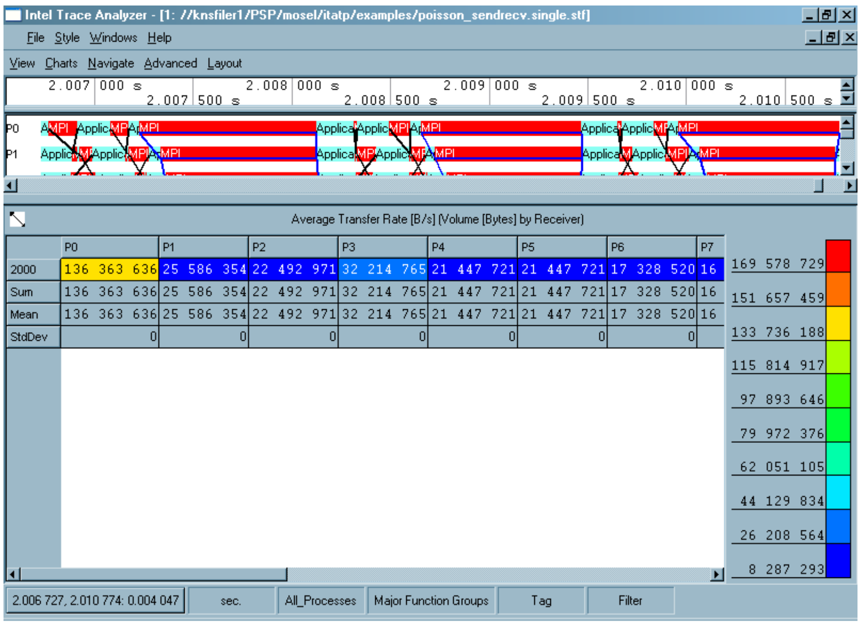
  Categorizes the messages by the function that sends them. Labels are names of MPI functions such as *MPI_Irsend*. This categorization is not influenced by the current Function Aggregation.

  > **Note:** This information is only available with 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.

  > **Note:** This information is only available with the Intel® Trace Collector 6 and higher.



	P0	P1	P2	P3	P4	P5	P6	P7
2000	136 363 636	25 586 354	22 492 971	32 214 765	21 447 721	21 447 721	17 328 520	16
Sum	136 363 636	25 586 354	22 492 971	32 214 765	21 447 721	21 447 721	17 328 520	16
Mean	136 363 636	25 586 354	22 492 971	32 214 765	21 447 721	21 447 721	17 328 520	16
StdDev	0	0	0	0	0	0	0	

Figure 4-36. Grouping Volume by Receiver

The Datum group allows choosing which attribute should be printed or painted in the cells. The available attributes are:

- *Total Time, [s] or [tick]*

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, [s] or [tick]*

  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, [s] or [tick]*

  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]*

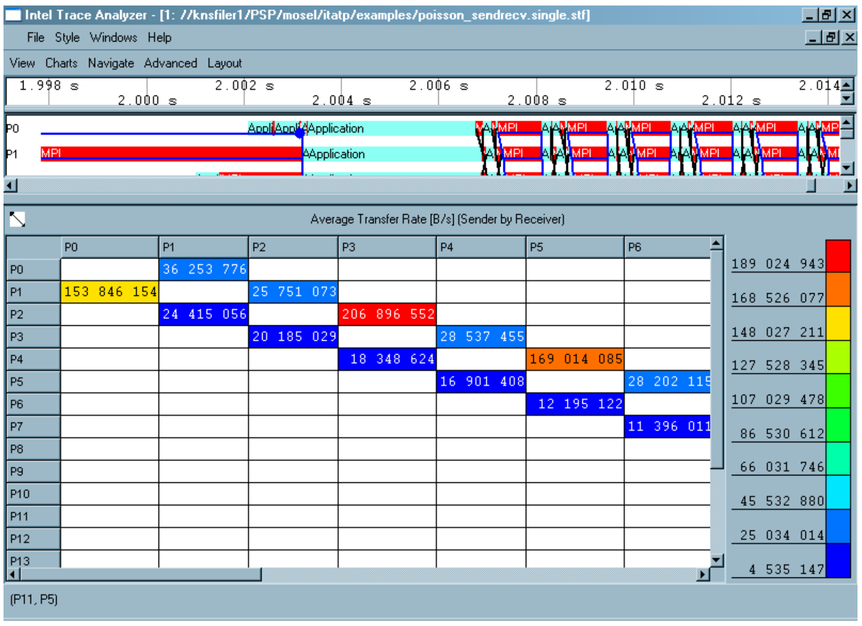
  The average transfer rate, averaged over the transfer rates of all messages that fall into this cell. Messages are not weighted; for example, transfer rates of short messages have the same impact as transfer rates of long messages.

- *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.


Intel Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_sendrecv.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
1.998 s
2.000 s
2.002 s
2.004 s
2.006 s
2.008 s
2.010 s
2.012 s
2.014
P0
P1
Appl
Appl
Application
MPI
Application
MPI

Average Transfer Rate [B/s] (Sender by Receiver)

	P0	P1	P2	P3	P4	P5	P6
P0		36 253 776					
P1	153 846 154		25 751 073				
P2		24 415 056		206 896 552			
P3			20 185 029		28 537 455		
P4				18 348 624		169 014 085	
P5					16 901 408		28 202 115
P6						12 195 122	
P7							11 396 011
P8							
P9							
P10							
P11							
P12							
P13							

189 024 943	🟥
168 526 077	🟧
148 027 211	🟨
127 528 345	🟩
107 029 478	🟩
86 530 612	🟩
66 031 746	🟩
45 532 880	🟦
25 034 014	🟦
4 535 147	🟦

(P11, P5)

Figure 4-37. Average Transfer Rate

- *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.

The group *Row Statistics* allows switching the individual columns on or off. These columns hold the statistics for the rows.

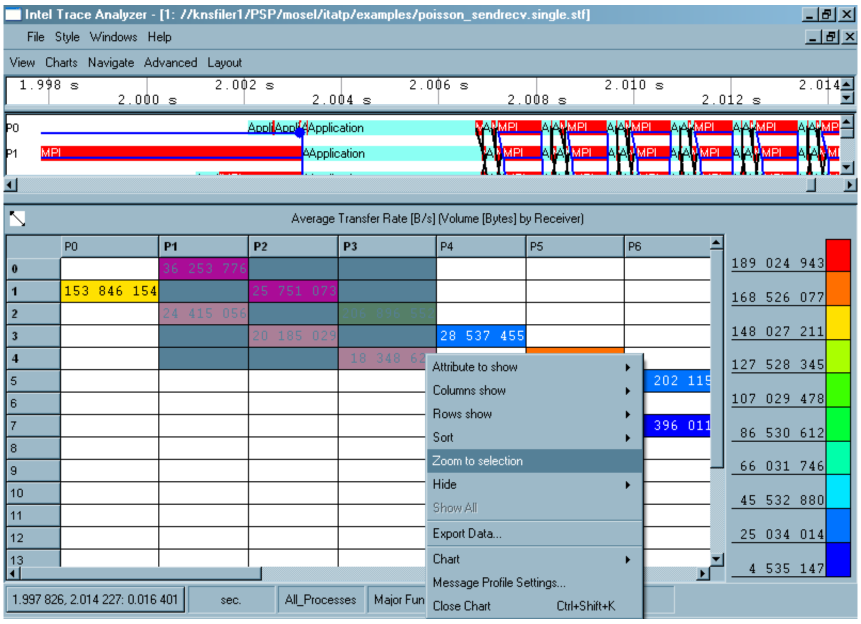
The group *Column Statistics* allows switching the individual rows on or off. These hold the statistics for the columns.

4.5.3. Context Menu

The context menu provides shortcuts with which the attributes and groupings are selected. To do this, use the entries *Attribute to show, Columns show* and *Rows show*. These entries are the same as those explained in Section 4.5.2 <(*Message Profile Settings dialog box*).

The entry sort allows to 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. Switching back to the default order is also useful if the columns or rows were rearranged by dragging the row or column headers around (hold the Ctrl key down while dragging to do that).

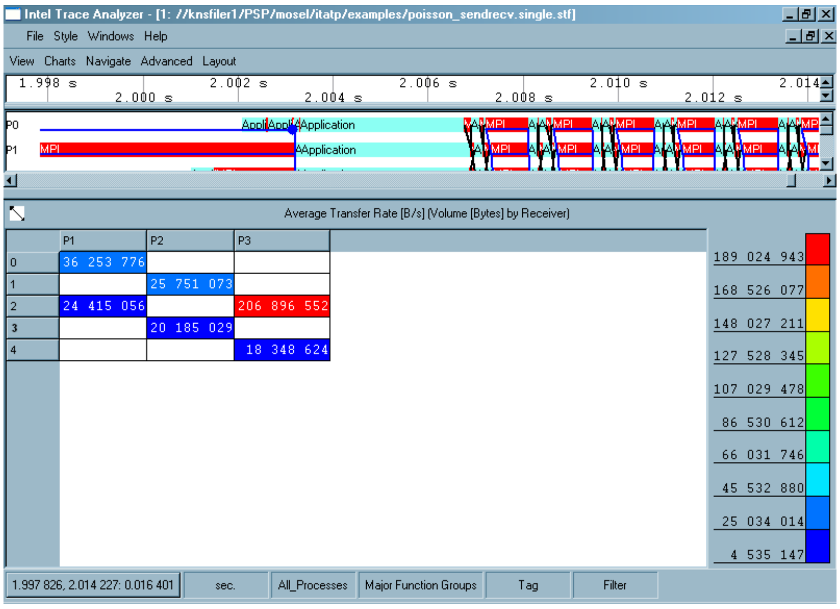
When a given area of the matrix is selected, then the context menu provides entries to either zoom into/out of the selected area or to suppress the display of the selected area (all row and columns that are partially selected are suppressed).



	P0	P1	P2	P3	P4	P5	P6
0		36 253 776					
1	153 846 154		25 751 073				
2		24 415 056		206 896 552			
3			20 185 029		28 537 455		
4				18 348 62			
5							202 115
6							
7							396 011
8							
9							
10							
11							
12							
13							

189 024 943	🟥
168 526 077	🟧
148 027 211	🟨
127 528 345	🟩
107 029 478	🟩
86 530 612	🟩
66 031 746	🟩
45 532 880	🟦
25 034 014	🟦
4 535 147	🟦

Figure 4-38. Selecting an area in the matrix and zooming into


Intel Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_sendrecv.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
1.998 s
2.000 s
2.002 s
2.004 s
2.006 s
2.008 s
2.010 s
2.012 s
2.014
P0
P1
Application
MPI

Average Transfer Rate [B/s] (Volume [Bytes] by Receiver)

	P1	P2	P3
0	36 253 776		
1		25 751 073	
2	24 415 056		206 896 552
3		20 185 029	
4			18 348 624

189 024 943	🟥
168 526 077	🟧
148 027 211	🟨
127 528 345	🟩
107 029 478	🟩
86 530 612	🟩
66 031 746	🟩
45 532 880	🟦
25 034 014	🟦
4 535 147	🟦

1.997 826, 2.014 227: 0.016 401 | sec. | All_Processes | Major Function Groups | Tag | Filter

Figure 4-39. Zoomed into the selected area

If something is hidden then the context menu provides an entry *Show All* and the *Hide* submenu contains enabled entries to unhide all hidden rows or columns or all.

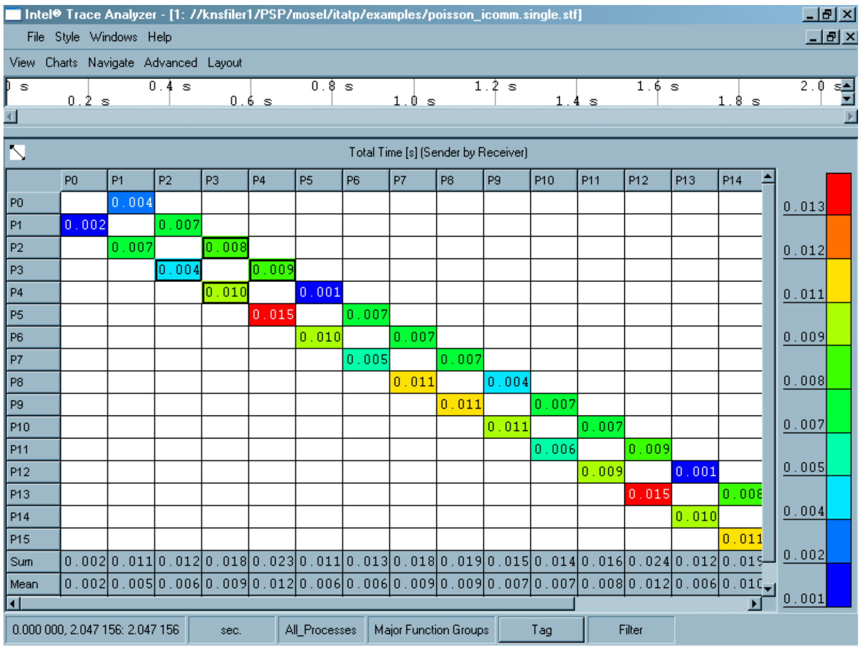
Actually the zoom feature of the Message profile relies on storing, which row and column labels are to be suppressed. This can have surprising effects: if *Volume* is selected as row grouping and the rows with labels 17 and 19 are hidden, then scrolling into an area containing messages with volume 18 results in these messages being shown. To suppress all messages with certain volumes, use filtering (see Section 5.1).

The entry *Export Data* opens a File Save dialog box to 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. It does not contain row or column statistics. For each cell, all available attributes are given.

Additionally the context menu contains the usual operations as described in Section 4.7.

4.5.4. 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 exists in that cell.


Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
0 s
0.2 s
0.4 s
0.6 s
0.8 s
1.0 s
1.2 s
1.4 s
1.6 s
1.8 s
2.0 s
0.013
0.012
0.011
0.009
0.008
0.007
0.005
0.004
0.002
0.001

Total Time [s] (Sender by Receiver)

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
P0		0.004													
P1	0.002		0.007												
P2		0.007		0.008											
P3			0.004		0.009										
P4				0.010		0.001									
P5					0.015		0.007								
P6						0.010		0.007							
P7							0.005		0.007						
P8								0.011		0.004					
P9									0.011		0.007				
P10										0.011		0.007			
P11											0.006		0.009		
P12												0.009		0.001	
P13													0.015		0.008
P14														0.010	
P15															0.011
Sum	0.002	0.011	0.012	0.018	0.023	0.011	0.013	0.018	0.019	0.015	0.014	0.016	0.024	0.012	0.019
Mean	0.002	0.005	0.006	0.009	0.012	0.006	0.006	0.009	0.009	0.007	0.007	0.008	0.012	0.006	0.010

0.000 000, 2.047 156: 2.047 156 | sec. | All_Processes | Major Function Groups | Tag | Filter

Figure 4-40. Tagging a process in the Message Profile

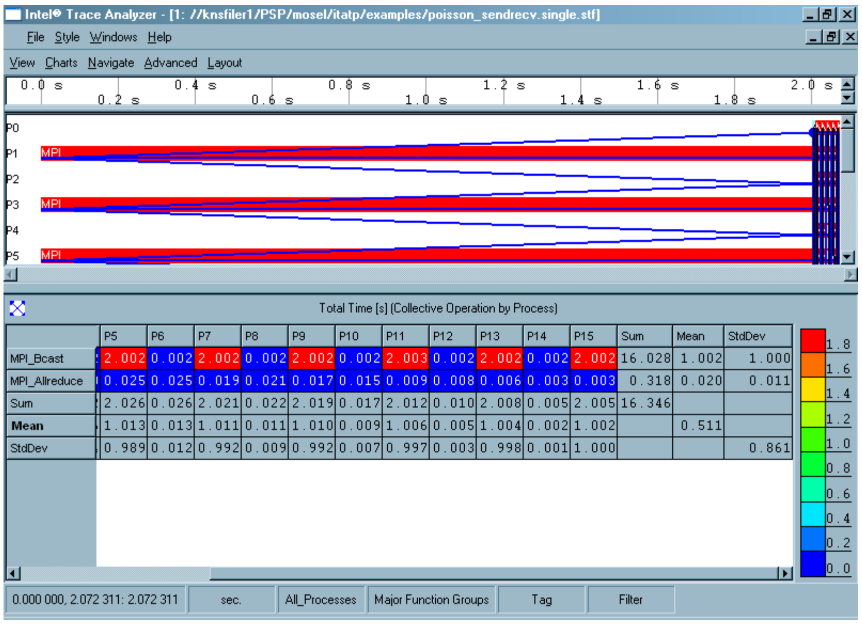
Messages that do not pass a filter are not accounted for and may result in a smaller matrix when this results in empty rows and columns. For more information on filtering and tagging, refer to Section 7.3.

4.5.5. Aggregation

The View's thread group influences the labels of the Sender, Receiver and Sender/Receiver groups. The View's 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.

4.6. The Collective Operations Profile

The Collective Operations Profile enables analyzing communication patterns that are done using MPI Collective Operations. Like the Message Profile, the Collective Operations is 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.


Intel® Trace Analyzer - [1: //knsfiler1/PSP/mosel/itatp/examples/poisson_sendrecv.single.stf]
File Style Windows Help
View Charts Navigate Advanced Layout
0.0 s 0.2 s 0.4 s 0.6 s 0.8 s 1.0 s 1.2 s 1.4 s 1.6 s 1.8 s 2.0 s
P0 P1 P2 P3 P4 P5
MPI
0.000 000, 2.072 311: 2.072 311
sec.
All_Processes
Major Function Groups
Tag
Filter
1.8 1.6 1.4 1.2 1.0 0.8 0.6 0.4 0.2 0.0

Total Time [s] (Collective Operation by Process)

	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	Sum	Mean	StdDev
MPI_Bcast	2.002	0.002	2.002	0.002	2.002	0.002	2.003	0.002	2.002	0.002	2.002	16.028	1.002	1.000
MPI_Allreduce	0.025	0.025	0.019	0.021	0.017	0.015	0.009	0.008	0.006	0.003	0.003	0.318	0.020	0.011
Sum	2.026	0.026	2.021	0.022	2.019	0.017	2.012	0.010	2.008	0.005	2.005	16.346		
Mean	1.013	0.013	1.011	0.011	1.010	0.009	1.006	0.005	1.004	0.002	1.002		0.511	
StdDev	0.989	0.012	0.992	0.009	0.992	0.007	0.997	0.003	0.998	0.001	1.000			0.861

Figure 4-41. Collective Operations Profile

4.6.1. Mouse Hover

When the mouse is positioned over any point in the matrix then detailed information for the current cell is shown in the View's status bar in the form "$AttributeValue ($RowLabel,$ColumnLabel) ". This allows getting exact attribute values even if the cells are configured to be very small or to show no alphanumerical entries at all.

4.6.2. Collective Operations Profile Settings

The Collective Operations Settings adjusts the various attributes that affect how the Chart is displayed. This includes the colors, the layout and the statistical attributes. The Settings dialog box is divided into three tabs namely the *Preferences* tab, the *Colors* tab and the *Data* tab.

- *The Preferences Tab*

  The *Preferences* tab adjusts the Display settings and the Layout settings.

  Display Group: In this group, the visual aspects of the Chart is configured. Using the check boxes *Row Labels* and *Column Labels*, it is decided if the row/column headers should be displayed or not. The check box *Scale*, if enabled, displays the colored scale that is seen on the right-hand side of the matrix. The *Grid* checkbox displays/removes the black grid in which the cells are placed. The checkbox *Keep Empty Rows/Columns when using Sender/Receiver Groupings* switches a special feature that is only relevant for the Groupings Sender and Receiver. For these groupings, a checked state of this box indicates that all processes should always be shown, like for example, showing even empty rows and columns. That keeps the form of the matrix constant and makes it easy to look for patterns in the data. An unchecked state means that 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.

  The radio buttons *Communicator Names* and *Communicator Ids* allow to either see helpful communicator names (if available in the trace file) that may take a lot of valuable screen space or to restrict the display to show only concise communicator ids.

Layout Group: The *Layout* group allows switching between two fundamentally different modes for the layout of the matrix. In the *Automatic Cell Sizes* mode, checking *Equal Cell Sizes* basically results in equal column widths and enables the check box *Square Cell Sizes* to get square cells. The other options of the *Layout group* are disabled by default.

Choosing *Manual Cell Sizes* allows specifying the size of the cells in pixels. 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 un-checking the check box *Text in Cells*.

- *The Colors Tab*

  The push buttons *Maximum Color* and *Minimum Color* allow choosing the colors for the maximum and minimum attribute values. The text input field allows specifying the number of color steps (1-255).

  The chosen colors are considered as points in a color space and the colors of the scale are interpolated on a line through color space connecting these two points. The combo box to the right of the text input field allows using either the HSV or the RGB color space. 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. Choosing white for the minimum and black for the maximum is not at all bad.

  By checking the box *Manual Scaling* it is possible to specify the minimum and maximum values for the color scale in the two text input fields below.

- *The Data Tab*

  The *Data* tab allows choosing how the data is analyzed. The *Data* tab is divided into the *grouping* section, the *Datum* section, the *Row Statistics* and the *Column Statistics*.

  Grouping Section: The *Grouping* section provides two combo boxes to choose the row and column headers or better said to choose how the data is grouped into categories. The groupings for rows and columns is chosen independently. However not all combinations are possible. It cannot have the same header for row and column. For example, a matrix cannot be plotted with both the row and column header being *Communicator*. All the headers are explained below:

  - *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 (see above) of the Settings dialog box.
  - *Collective Operation*: This shows the types of operations like MPI_Allreduce and MPI_Bcast.
  - *Root*: Shows the root used in the operation, if applicable. If there is no root, a label *No Root* is created.
  - *Process*: Categorizes the operations by the processes.

  Datum Section: The *Datum* group allows choosing which attribute should be printed or painted in the cells. The available attributes are:

  - *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: Under *Row Statistics*, it is specified 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.

4.6.3. The Context Menu

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

- *Attribute to show*

  The attributes to be shown in the Collective Operations Profile is selected in this option. It contains all the attributes that are explained in the Datum Section.

- *Columns to show*

  This entry indicates if the Collective Operations Profile should be displayed by process, by root or by communicator.

- *Rows to show*

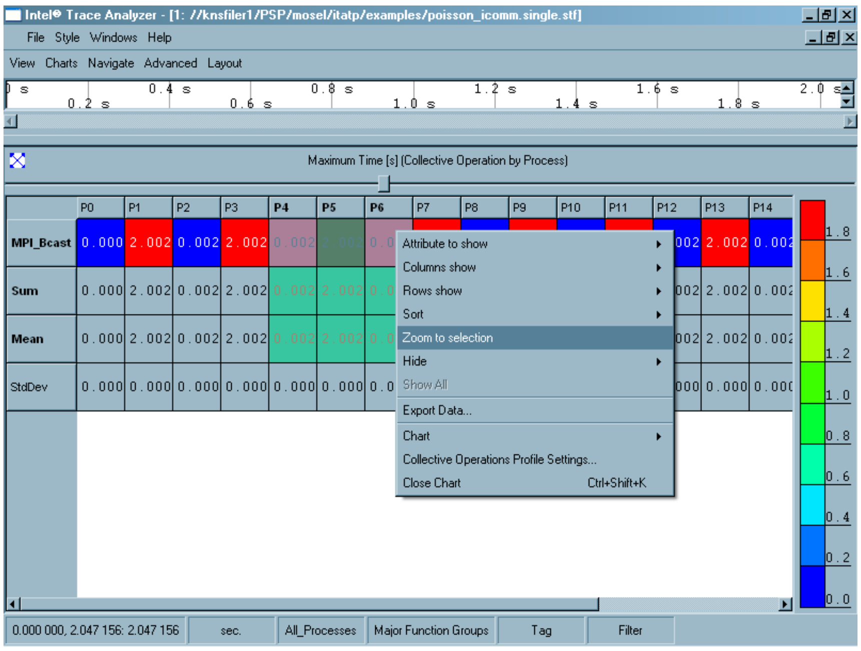
  This entry denotes whether the rows of the profile show the Collective Operation, the communicator or the root values.

- *Sort*

  The entry sort enables sorting 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. Switching back to the default order is also useful if the columns or rows were rearranged by dragging the row or column headers around (hold the Ctrl key down while dragging).

- *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 as shown in Figure 4-42 and choose the entry *Zoom to selection* from the context menu (obtained by a right-click on the mouse). As a result, everything else other than the selected region, is removed from the display.



	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
MPI_Bcast	0.000	2.002	0.002	2.002	0.002	2.002	0.0						002	2.002	0.002
Sum	0.000	2.002	0.002	2.002	0.002	2.002	0.0						002	2.002	0.002
Mean	0.000	2.002	0.002	2.002	0.002	2.002	0.0						002	2.002	0.002
StdDev	0.000	0.000	0.000	0.000	0.000	0.000	0.0						000	0.000	0.000

Figure 4-42. Zoom to Selection in the Collective Operations Profile

- *Hide*

  This hides all cells that are selected. Selection of cells is done by holding down the left mouse button and moving over the required region. This also automatically opens the context menu.

- *Show All*

  This entry shows all cells again. It is enabled only if cells have been previously hidden using the *Hide* entry.

- *Export Data*

  This entry opens a File Save dialog box to 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 Settings Profile*

  This opens the Settings dialog box of the Collective Operations Profile.

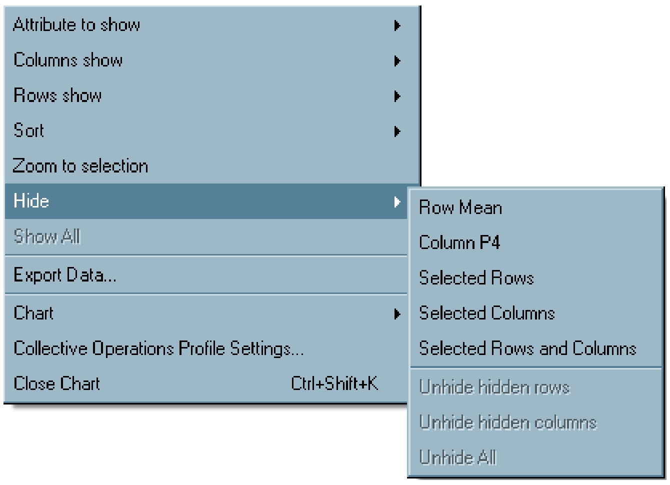


Figure 4-43. Context menu of the Collective Operations Profile

4.6.4. 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. For more information on tagging and filtering, refer to Section 7.3.

4.7. Common Chart Features

All Charts share some common features that are available via common context menu entries:

- *Print Chart (Ctrl+Shift+P)*

  Prints a hard copy of the Chart.

- *Save Chart (Ctrl+Shift+S)*

  Saves the chart as a picture.

- *Clone Chart (Ctrl+Shift+D)*

  Opens a new View containing exactly the same Charts as in the present View. In other words, it “clones” the current View.

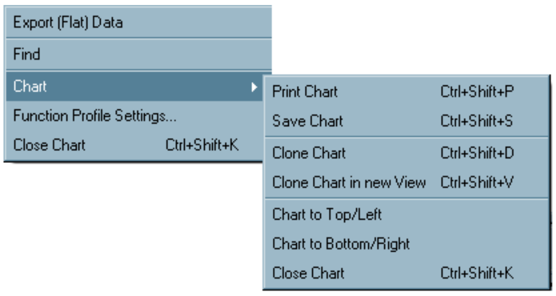


Figure 4-44. Common context menu features

- *Clone Chart in New View (Ctrl+Shift+V)*

  Makes a clone of the Chart in a new View

- *Chart to Top/Left*

  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, then this entry moves the selected Chart to the top. If they are placed next to each other, then it moves the selected Chart to the left.

- *Chart to Bottom/Right*

  This entry is also only 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, then this entry moves the selected Chart to the bottom. If they are placed next to each other, then it moves the selected Chart to the right.

- *Close Chart (Ctrl+Shift+K)*

  Closes the Chart.

When a Function Group A is right-clicked then all Charts show the entry **Context Menu**⟶**Ungroup A**. When a child of a recently ungrouped Function Group A′ is right-clicked then all Charts show the entry **Context Menu**⟶**Regroup A** .

To move the Chart to different positions in the View, use **Context Menu**⟶**Chart**⟶**Chart to Top/Left** or **Context Menu**⟶**Chart**⟶**Chart to Bottom/Right**.

Chapter 5. 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 that the dialog boxes current settings are inconsistent or out of bounds, the *OK* and *Apply* buttons are both disabled.

5.1. The Filtering Dialog

The filtering dialog box is accessed through the Advanced Menu (**Views Menu**⟶**Advanced**⟶**Filtering**) This dialog box allows specifying filter expressions that describe which function events, messages and collective operations are to be analyzed and shown. The two radio buttons in the group *Definition of Filter Expression* at the top switch between two fundamental modes: *Using GUI Interface* is chosen by default and allows generating the filter expression via a graphical interface, while *Manually* allows to type in the filter expression directly. Building the expression with the graphical interface is a lot easier and recommended for new or infrequent users.

An expression is built using either the point and click interface or using the manual mode. Either way the resulting expression is parsed upon each change.

If the current expression can not be converted into a proper filter definition then the dialog shows a red warning that indicates the reason.

If the expression makes use of filter attributes that require to bypass the in-memory cache and to process detailed data from the trace file then a yellow warning is shown. In this case it can be expected that the analysis using this filter expression will need considerably more time than usual.

5.1.1. Building Filter Expressions Using the Graphical Interface

The Filtering dialog box allows specifying filter expressions separately for function events, messages and collective operations via three separate tabs. It provides a fourth tab labeled *Processes* to further restrict the events to only 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`** are described.

- *Processes Tab*

  This tab specifies which processes should pass the filter. It has effects on all three sub-expressions.

  The *Restrict Filtering to Processes* text field defines which processes should be taken into account and be filtered further according to the settings of the other three tabs. Processes not listed in the text field are filtered out completely unless the text field is empty: this is the default and indicates that the *Processes* tab has no effect. The text field can take 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.

  The button labeled ... opens the dialog box *Process Group Selection* where one or more processes or process groups are selected (see below). The *Invert* check box swaps the selection. For example, if all processes except one should pass the filter, select the process to be filtered out and use the check box *Invert*.

  Checking *Invert* means to let only events pass that do not match this predicate. It is a logical NOT and shown as a exclamation mark before the predicate in the filter expression.

Below the filter clauses, the resulting filter expression is shown. To reuse the expression elsewhere, select and copy the expression in the manual mode of the filtering dialog box. The selection is done using the mouse. The context menu of the filter expression contains a *Copy (Ctrl+C)* entry and a *Select All (Ctrl+A)* entry.

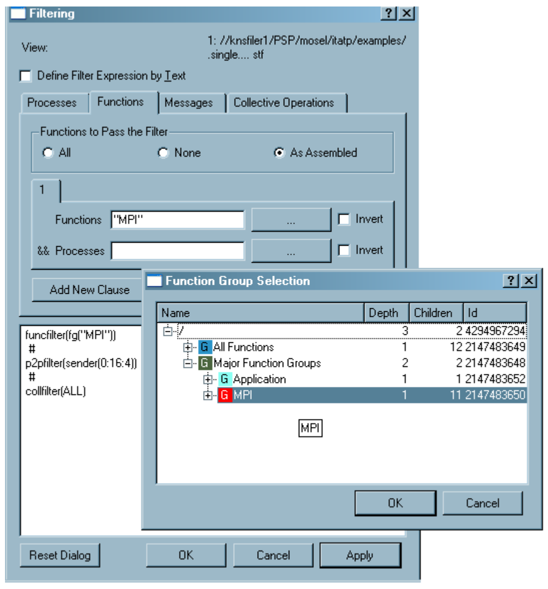
- *Functions*

  In the *Functions* tab, the mode of filtering is selected by means of radio buttons - *All*, *None* and *Custom*.

  Again, on selecting the *Custom* radio button, the filter clause tab is enabled. This consists of the following entries:

  Functions: The text field can take 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. Names given match all equally named functions/groups. The button ... opens the *Function Group Selection* subdialog which allows choosing the function names from a list. In addition, the subdialog provides a checkbox *Add Function Id*: if checked, not only is the name of a selected function written into the text field, but also its Id. 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: The text field can take 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. Names given match all equally named processes/groups. The button ... allows choosing from a list. The corresponding *Process Group Selection* subdialog provides a checkbox *Add Process Id* which is similar to the above *Function Group Selection* subdialog.



View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.... stf

Functions to Pass the Filter: All / None / [✓ As Assembled]

Functions "MPI"

&& Processes

```
funcfilter(fg("MPI"))
 #
p2pfilter(sender(0:16:4))
 #
collfilter(ALL)
```

Name	Depth	Children	Id
/	3	2	4294967294
All Functions	1	12	2147483649
Major Function Groups	2	2	2147483648
Application	1	1	2147483652
MPI	1	11	2147483650

Figure 5-1. Function group selection opened via the filter dialog box

The button *Add New Clause* specifies another filter clause. 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; for example, they form a so-called *And Clause*.

- *Messages*

  On selecting the *Custom* radio button (see Figure 5-1), the filter clause tab in the Messages tab is enabled. It has the following entries:

  Communicator: The text field can take a comma-separated list of communicator ids, unquoted communicator names or communicator names in double quotes that pass the filter. The button ... allows choosing from a list.

  Tag: The text field can take a comma-separated list of non-negative integers and triplets that describe tag values that pass the filter.

  Message Size: The text field can take a comma-separated list of non-negative integers and triplets that describe message sizes (in bytes) that pass the filter.

  Sender: The text field can take 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. Names given match all equally named processess/groups. The button ... allows choosing from a list.

  Receiver: Analogous to Sender.

  Processes: Makes a message pass the filter if either the sender or the receiver matches. Analogous to the logical OR of Sender and Receiver.

  Ranks of Sender: The text field can take a comma-separated list of non-negative integers and triplets that describe sender ranks (in the MPI communicator) that can pass the filter.

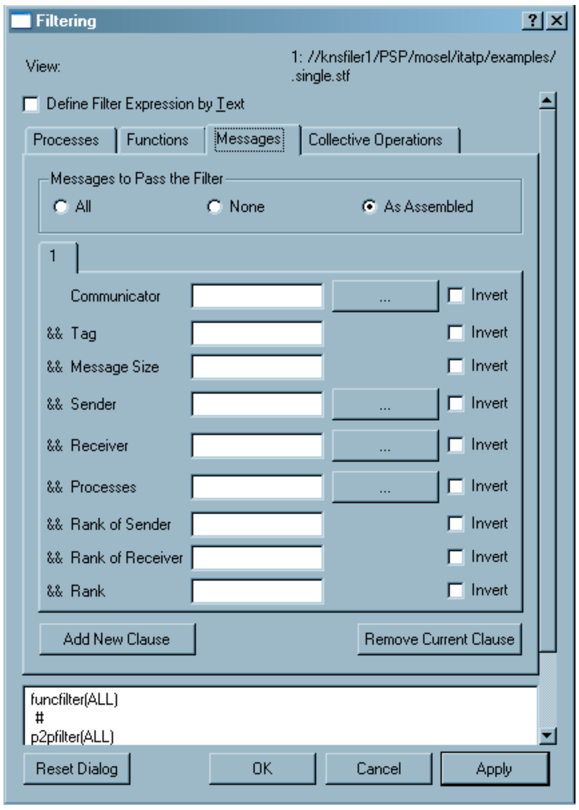
  Ranks of Receiver: Analogous to Rank of Sender.

  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.

  Start Time: The text field can take a comma-separated list of non-negative integers and triplets that describe start time (in ticks) of the message that make the operation pass the filter. The button ... allows to enter/edit the time in ticks or seconds (default depending on the View's current time unit).

  End Time: Analogous to Start Time.

  Duration: The text field can take a comma-separated list of non-negative integers and triplets that describe the duration (in ticks) of the message that make the operation pass the filter. The button ... allows to enter/edit the duration (shown as a time interval) in ticks or seconds (default depending on the View's current time unit).



Messages to Pass the Filter: All [✓ As Assembled] None

```
funcfilter(ALL)
 #
p2pfilter(ALL)
```

Figure 5-2. The filtering dialog box showing the Messages tab

- *Collective Operations*

  On selecting the *Custom* radio button the filter clause tab in the Messages tab is enabled. It has the following entries:

  Communicator: The text field can take a comma-separated list of communicator ids, unquoted communicator names or communicator names in double quotes that pass the filter. The button ... allows choosing from a list.

  Collective Operation: The text field can take a comma-separated list of unquoted or double-quoted names of collective operations like "MPI_Allreduce" that pass the filter. The button ... allows choosing from a list.

  Transferred Volume: The text field can take a comma-separated list of non-negative integers and triplets that describe all volumes (in total bytes per operation) that make a collective operation make pass the filter.

  Processes: The text field can take 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 participating in the operation that make the operation pass the filter. Names given match all equally named processes/groups. The button ... allows choosing from a list.

  Root: The text field can take 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 make the operation pass the filter. Names given match all equally named processes/groups. The button ... allows choosing from a list.

  Rank of Root: The text field can take a comma-separated list of non-negative integers and triplets that describe root ranks of the operation that make the operation pass the filter.

  Start Time: The text field can take a comma-separated list of non-negative integers and triplets that describe start time (in ticks) of the operation that make the operation pass the filter. The button ... allows to enter/edit the time in ticks or seconds (default depending on the View's current time unit).

End Time: Analogous to Start Time.

Duration: The text field can take a comma-separated list of non-negative integers and triplets that describe the duration (in ticks) of the operation that make the operation pass the filter. The button ... allows to enter/edit the duration (shown as a time interval) in ticks or seconds (default depending on the View's current time unit).

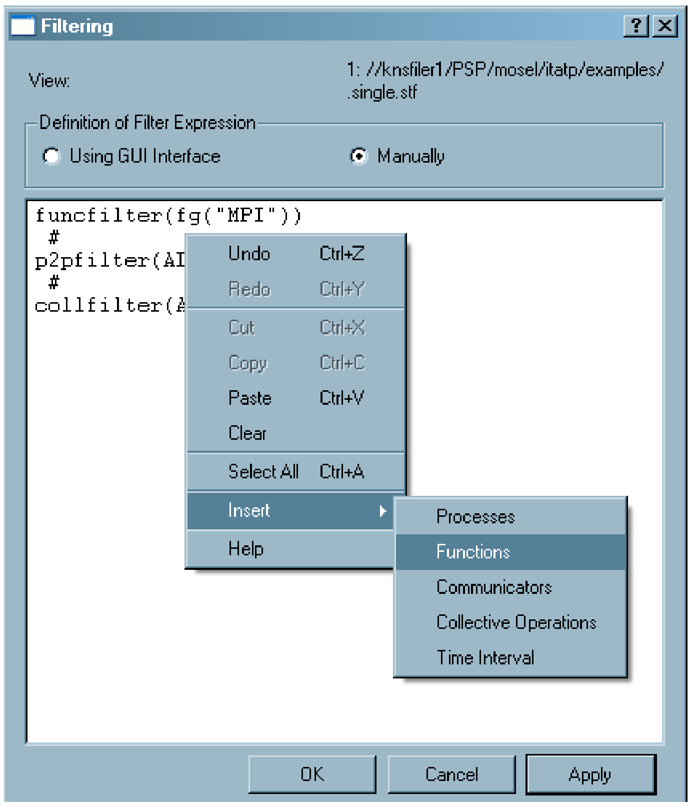
5.1.2. Building Filter Expressions Manually

Manual mode allows constructing any filter expression that is valid as described by the expression grammar in Section 5.1.3.

Note that for convenience there are context menu entries that allow to select processes, functions, communicators and collective operations from a dialog box in the same way as from the point and click 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, use parentheses if needed.



```
funcfilter(fg("MPI"))
 #
p2pfilter(AI
 #
collfilter(A
```

Figure 5-3. The Filtering dialog box in manual mode

5.1.3. The Filter Expression Grammar

The filter expression grammar creates a filter, which 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 a # and come in any order. Each filter class is specified once, more than once (in which 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 by 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 Section 5.1.1. (See a formal description of the grammar in Section 5.1.3.1.) 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 allows defining a special expression for a filter that lets all or no data of that kind pass through it. For example, **`p2pfilter(NONE)`** filters out all messages, while **`collfilter(ALL)`** lets all collective operations pass. When **`ALL`** or **`NONE`** keyword is used, it must be the only argument to **`funcfilter`**, **`p2pfilter`**, or **`collfilter`**.

Keyword specification in the filter expression grammar is case-insensitive. Specifying names (for functions, processes, communicators, etc.), however, is case-sensitive. Double quotes are needed 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 then it is evaluated as if all matching groups were given.

5.1.3.1. A Formal Description of the 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
        | ( 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 )

# 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 semicol

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]+
```

5.1.3.2. Examples of Advanced Usage of the Grammar

This section includes several examples of manually using the filter expression grammar and how it provides advanced capabilities in filtering trace data and speeding up the process of selecting exactly what the user would like to be analyzed.

For the first example, consider a parenthesized structure that can not be built by the point and click interface as 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 predicate **`sr`**, which is not available in the point and click interface, to efficiently filter out all messages between processes 0 and 1:

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

Finally, consider the following scenario. With the point and click interface, a complicated filter is specified for a certain filter class with a large number of predicates and Boolean oprators (both AND and OR, the latter added by using the *Add New Clause* button). Now, 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 ) )
```

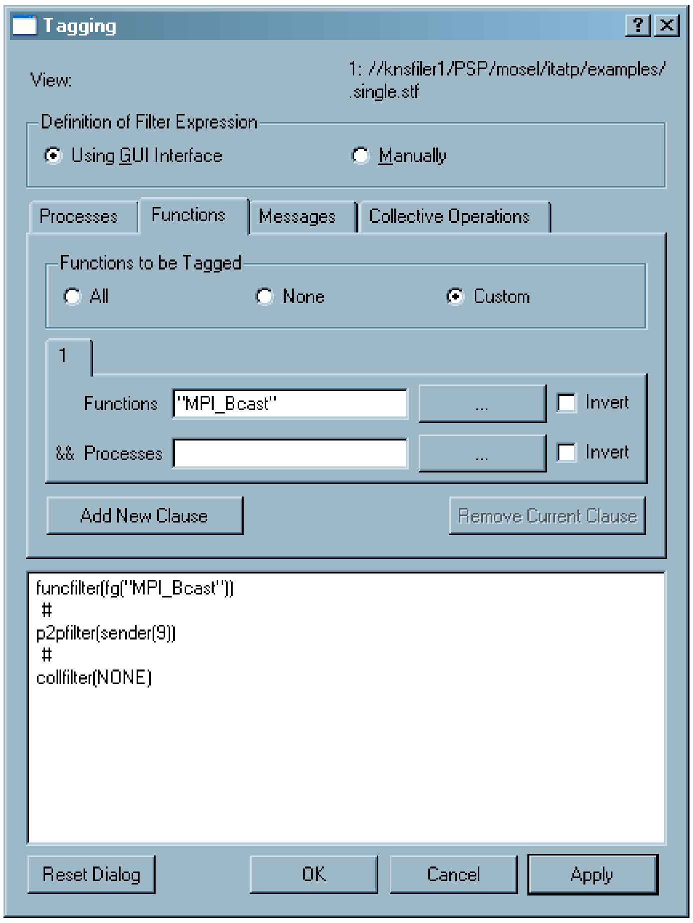
while this filter specifies all the collective operations that do not match the description above:

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

5.2. The Tagging Dialog

For an explanation of the filtering concept refer to Section 7.3. The usage of this dialog box is as for the Filtering dialog box and described in Section 5.1.

The only difference is that the default Filtering dialog box lets all events pass while the default Tagging dialog box does not tag any event at all.



View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.stf

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

Functions to be Tagged: All None [✓ Custom]

Functions "MPI_Bcast" Invert [ ]

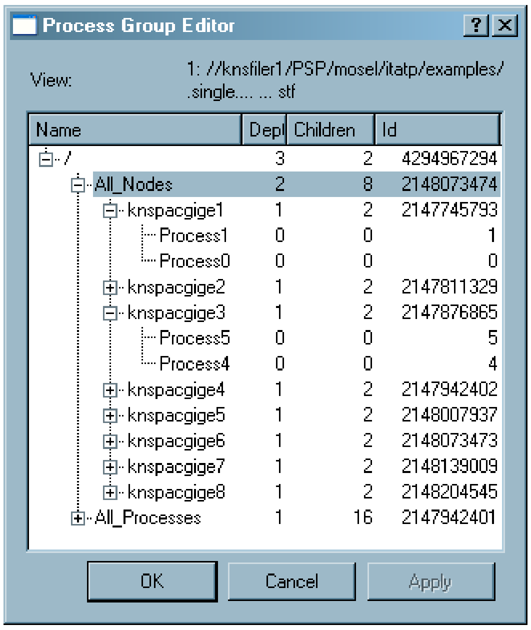
&& Processes Invert [ ]

```
funcfilter(fg("MPI_Bcast"))
 #
p2pfilter(sender(9))
 #
collfilter(NONE)
```

Figure 5-4. The Tagging Dialog

5.3. The Process Group Editor

The Process Group Editor is found at (**Advanced**⟶**Process Aggregation**. The Process Group Editor provides two functions. One is to select a process group (or function group in general) for process aggregation. The other is to create new groups beyond the ones that are provided by default. Group definitions are stored in the file .itarc in the user's home directory (for an English Microsoft* Windows* XP* installation this should be something like `C:\Documents and Settings\%username%\.itarc`). Refer to Section 5.10 for other ways to save, edit and load configuration information.



View: 1: //knsfiler1/PSP/mosel/itatp/examples/ .single.... ... stf

Name	Depl	Children	Id
/	3	2	4294967294
All_Nodes	2	8	2148073474
knspacgige1	1	2	2147745793
Process1	0	0	1
Process0	0	0	0
knspacgige2	1	2	2147811329
knspacgige3	1	2	2147876865
Process5	0	0	5
Process4	0	0	4
knspacgige4	1	2	2147942402
knspacgige5	1	2	2148007937
knspacgige6	1	2	2148073473
knspacgige7	1	2	2148139009
knspacgige8	1	2	2148204545
All_Processes	1	16	2147942401

Figure 5-5. The Process Group Editor

To select a process group for aggregation (see Section 7.2 for an explanation of the concept of Aggregation) select the group by using the mouse or the cursor/arrow keys and press the *Apply* button or the *OK* button. Note that the dialog box only accepts a single, non-empty process group that contains each of its functions no more than once.

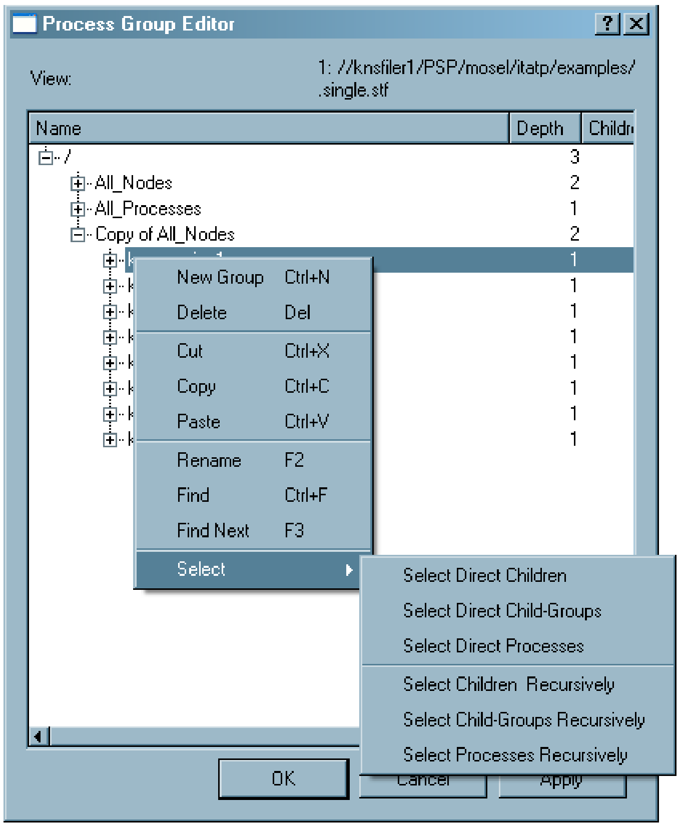


Figure 5-6. The Process Group Editor's context menu

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

The entry *New Group* creates a new group as a child of the clicked item. The entry *Delete* removes an item. This entry is disabled for items that originate of the trace file. Only user-created items are deleted.

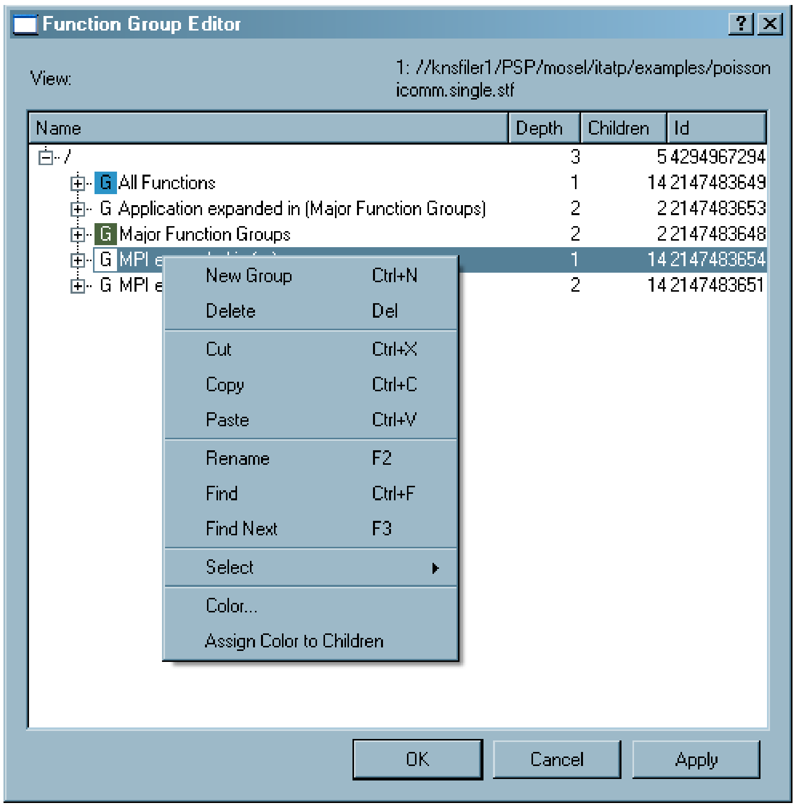
The entry *Rename (F2)* allows to rename user-created items. The entry *Find (Ctrl+F)* opens the Find dialog box (see Section 5.11) and *Find Next (F3)* searches for the next match if a search was started before.

The entry *Select* opens a submenu that allows conveniently selecting 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.

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 results in groups that contain the same function twice or more often. Such groups are accepted for storage but not for aggregation. Empty groups are deleted automatically when the dialog box is closed.

5.4. The Function Group Editor

The Function Group Editor is accessed at **Advanced**⟶**Function Aggregation**. In most respects the Function Group Editor works exactly like the Process Group Editor (see Section 5.3). It allows to edit function group definitions and to choose a function group for aggregation in a View. The only addition is that functions and function groups are assigned colors via the context menu. (see Section 5.5).



View: 1: //knsfiler1/PSP/mosel/itatp/examples/poisson icomm.single.stf

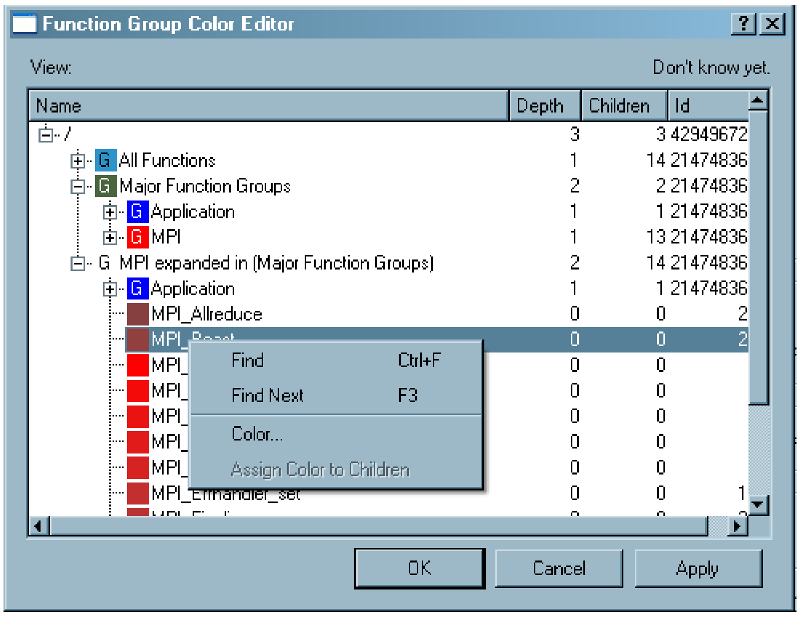
Name	Depth	Children	Id
/	3	5	4294967294
G All Functions	1	14	2147483649
G Application expanded in (Major Function Groups)	2	2	2147483653
G Major Function Groups	2	2	2147483648
G MPI e	1	14	2147483654
G MPI e	2	14	2147483651

Figure 5-7. The Function Group Editor's context menu

To change the color of any of the functions, select the *Color* option from the context menu shown in Figure 5-7. A Color dialog box is opened where the preferred color for the given function is selected. The context menu also has the option of assigning the color of the parent node to all the children, using the *Assign Color to Children* entry. This entry is enabled only if a parent node is selected.

5.5. The Function Group Color Editor

This is the Function Group Editor in a restricted mode that only allows editing the colors of functions and function groups. The Function Group Color Editor can only be accessed when choosing to change the function colors in the Event Timelines settings dialog box (see Section 4.1). The context menu of the *Function Group Color Editor* has a *Find* and a *Find Next* entry, both of which are explained in Section 5.3 and also a *Color* entry and a *Assign Color to Children* entry, which are explained in Section 5.4.


Function Group Color Editor

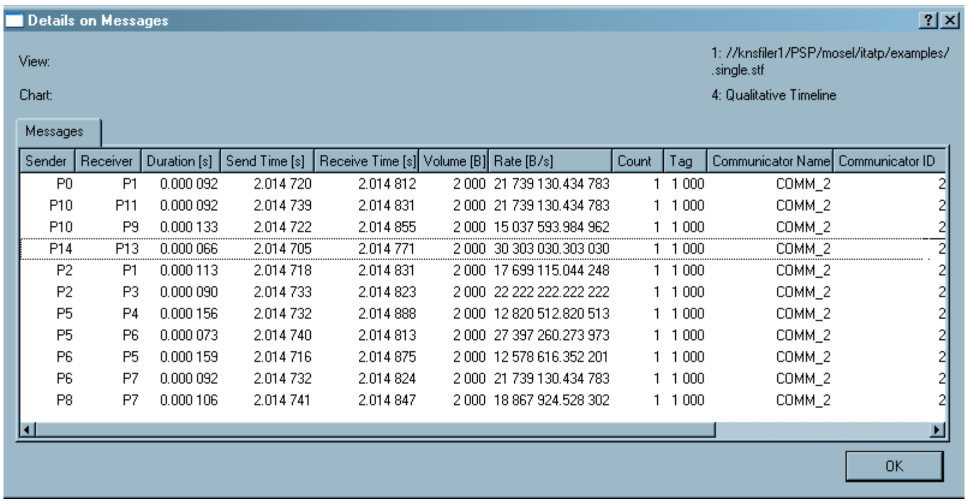
View: Don't know yet.

Name	Depth	Children	Id
/	3	3	42949672
All Functions	1	14	21474836
Major Function Groups	2	2	21474836
Application	1	1	21474836
MPI	1	13	21474836
MPI expanded in (Major Function Groups)	2	14	21474836
Application	1	1	21474836
MPI_Allreduce	0	0	2
MPI_	0	0	2
MPI_	0	0	
MPI_	0	0	
MPI_	0	0	
MPI_	0	0	
MPI_	0	0	
MPI_Errhandler_set	0	0	1

Figure 5-8. The Function Group Color Editor

5.6. The Details Dialog

This dialog box is available from the context menus of the Event Timeline (see Section 4.1) and the Qualitative Timeline (see Section 4.2).


Details on Messages

View: 1: //knsfiler1/PSP/mosel/itatp/examples/.single.stf

Chart: 4: Qualitative Timeline

Messages

Sender	Receiver	Duration [s]	Send Time [s]	Receive Time [s]	Volume [B]	Rate [B/s]	Count	Tag	Communicator Name	Communicator ID
P0	P1	0.000 092	2.014 720	2.014 812	2 000	21 739 130.434 783	1	1 000	COMM_2	2
P10	P11	0.000 092	2.014 739	2.014 831	2 000	21 739 130.434 783	1	1 000	COMM_2	2
P10	P9	0.000 133	2.014 722	2.014 855	2 000	15 037 593.984 962	1	1 000	COMM_2	2
P14	P13	0.000 066	2.014 705	2.014 771	2 000	30 303 030.303 030	1	1 000	COMM_2	2
P2	P1	0.000 113	2.014 718	2.014 831	2 000	17 699 115.044 248	1	1 000	COMM_2	2
P2	P3	0.000 090	2.014 733	2.014 823	2 000	22 222 222.222 222	1	1 000	COMM_2	2
P5	P4	0.000 156	2.014 732	2.014 888	2 000	12 820 512.820 513	1	1 000	COMM_2	2
P5	P6	0.000 073	2.014 740	2.014 813	2 000	27 397 260.273 973	1	1 000	COMM_2	2
P6	P5	0.000 159	2.014 716	2.014 875	2 000	12 578 616.352 201	1	1 000	COMM_2	2
P6	P7	0.000 092	2.014 732	2.014 824	2 000	21 739 130.434 783	1	1 000	COMM_2	2
P8	P7	0.000 106	2.014 741	2.014 847	2 000	18 867 924.528 302	1	1 000	COMM_2	2

Figure 5-9. Details on Messages shown in the Qualitative Timeline

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.

If the column *Count* shows a value greater than one, the event was created by *merging* several atomic events (see Section 7.1). 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 to filter out the other event categories in the Detail View to be opened. Note that a left click on reuses an existing Detail View so that the screen is not cluttered so easily. If desired, right-clicking the drill down button opens 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 (see Section 5.7). Note that the source code location is only available upto 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.

5.6.1. Detailed Attributes of Function Events

- *Name:* This attribute specifies the name of the selected function. If it represents a Function Group, then it is prefixed with the Group name.
- *Process Group:* Specifies the Process, Function or Group in which the selected event occurred
- *Duration:* This indicates 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:* This shows 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:* This shows 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.

  Note: Sometimes a Function extends beyond the displayed time interval. If a Function begins or ends outside the current zoom interval, then in the *Details* dialog the boundary of the current zoom interval is used as the *Start* and/or *End* time of the given function.

- *Count*: The attribute indicates the number of function calls in a selected time interval. If the count is greater than 1 the selection represents more than a single function event.

5.6.2. Detailed Attributes of Message Events

- *Sender:* The Sender is the Process, Function or Group which sent the message.
- *Receiver:* Process, Function or Group which received the message.
- *Duration:* The 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:* This indicates the rate at which the bytes are transferred. It is calculated using *Volume/Duration*.
- *Count:* This specifies the number of messages that are merged into the selection.

- *Tag:* This attribute 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:* This specifies the name of the MPI function which received the sent message(s).

5.6.3. Detailed Attributes of Collective Operation Events

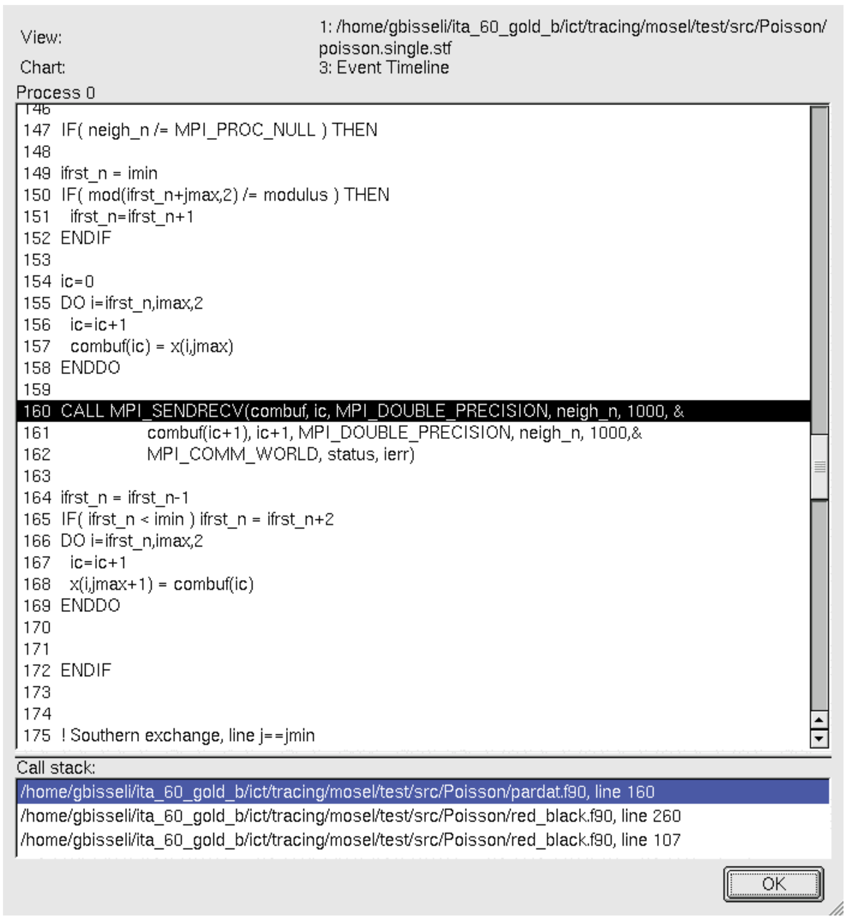
Each possibly merged collective operation has a header entry which describes the collective operation as a whole. The plus handle gets a detailed list of the same information per Process/Group. The exact processes or process groups shown depend on the current process aggregation.

- *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.

5.7. Source View Dialog

The Source View dialog box is opened from the Details dialog box (see Section 5.6) if source code location information is available in the trace file.

The dialog box consists of a combo box to choose processes, a text browser in the center and a list box representing the call stack at the bottom.



View: 1: /home/gbisseli/ita_60_gold_b/ict/tracing/mosel/test/src/Poisson/poisson.single.stf
Chart: 3: Event Timeline
Process 0

```
146
147  IF( neigh_n /= MPI_PROC_NULL ) THEN
148
149  ifrst_n = imin
150  IF( mod(ifrst_n+jmax,2) /= modulus ) THEN
151    ifrst_n=ifrst_n+1
152  ENDIF
153
154  ic=0
155  DO i=ifrst_n,imax,2
156    ic=ic+1
157    combuf(ic) = x(i,jmax)
158  ENDDO
159
160  CALL MPI_SENDRECV(combuf, ic, MPI_DOUBLE_PRECISION, neigh_n, 1000, &
161                    combuf(ic+1), ic+1, MPI_DOUBLE_PRECISION, neigh_n, 1000,&
162                    MPI_COMM_WORLD, status, ierr)
163
164  ifrst_n = ifrst_n-1
165  IF( ifrst_n < imin ) ifrst_n = ifrst_n+2
166  DO i=ifrst_n,imax,2
167    ic=ic+1
168    x(i,jmax+1) = combuf(ic)
169  ENDDO
170
171
172  ENDIF
173
174
175  ! Southern exchange, line j==jmin
```

Call stack:

```
/home/gbisseli/ita_60_gold_b/ict/tracing/mosel/test/src/Poisson/pardat.f90, line 160
/home/gbisseli/ita_60_gold_b/ict/tracing/mosel/test/src/Poisson/red_black.f90, line 260
/home/gbisseli/ita_60_gold_b/ict/tracing/mosel/test/src/Poisson/red_black.f90, line 107
```

Figure 5-10. The Source View dialog box

The combo box allows selecting from several processes (functions) if the dialog box was opened for a collective operation, from two processes if it was opened for a message and it degenerates to a label if the dialog box was opened for a function event.

The text browser shows a source file. The line corresponding to the current stack level is shown highlighted in reverse video.

The list at the bottom allows to 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.

The default is to search source code files in the current directory and the directory of the current trace file. Use the entry *File Default/UserDefines SCLSearchPaths* in the configuration dialog box to specify additional directories to be searched. Refer to Section 5.10.1 for details. If no source files are found, then a file open dialog box is shown to manually specify the source file to load.

5.8. Time Interval Selection

When opened from **Views Menu**⟶**Navigate**⟶**Goto (G)**, this dialog box allows entering a new time interval for the whole View. This interval is pushed onto the zoom stack (see Section 3.3.1) and the View is updated accordingly.

When opened from the filter dialog boxes (see Section 5.1 and Section 5.2), this dialog box enters or edits a time interval or duration in a filter expression.

The time interval is specified in ticks or seconds. The interval is entered either by giving the start and stop or by giving the center and width. Entering a value that is greater than the maximum value of the trace file's time interval is possible; this value is automatically reduced to the maximum value (*tmax* of the trace).

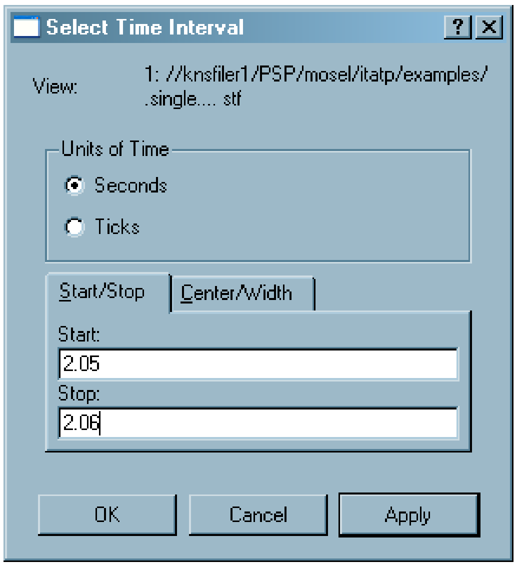


Figure 5-11. Selecting a time interval

5.9. The New View Dialog

This dialog box appears when **Views Menu**⟶**View**⟶**New View** is chosen and allows specifying which charts should be shown in the new View.

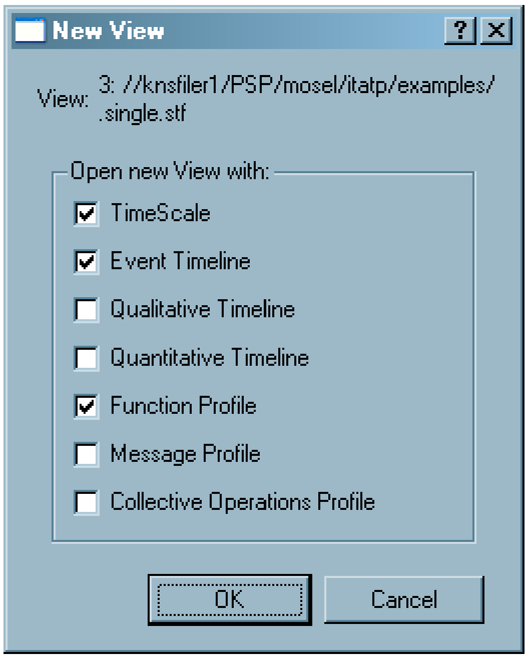


Figure 5-12. The New View dialog box

5.10. The Configuration Dialogs

These dialog boxes enable manipulating the configuration that is usually stored in the file `.itarc` in the user's home directory upon program exit. The configuration consists of the global information found under the option *File Default* and the per file information found under the respective file name.

The global information consists of the recent file list and the list of search paths finds source code files. The latter is stored in the option *File Default/UserDefines SCLSearchPaths* and should contain a list of directories, separated by semi-colons (`;`), that are searched in the given order for source code files to be shown in the Source View dialog box (see Section 5.7).

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

5.10.1. The Edit Configuration Dialog

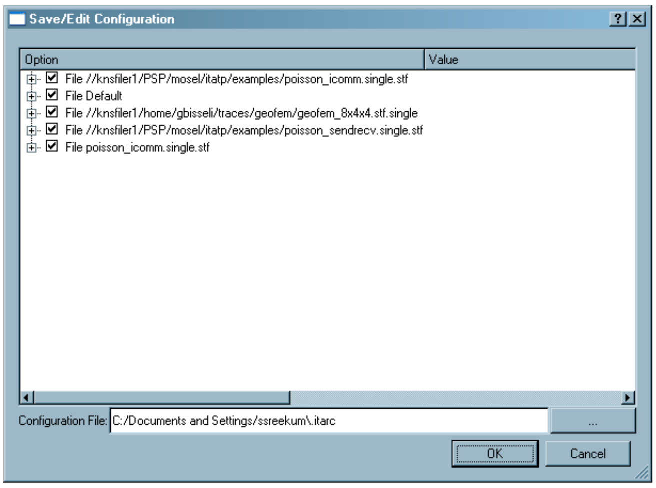
This dialog box allows editing the configuration; for example, by changing values or removing entries from the configuration.

This dialog box together with the Load Configuration dialog box allows to store per trace file settings together with the trace file so that definitions of process groups and function groups are shared in a work group.

The entries are shown with check box like handles that are all checked by default. Unchecking entries removes them from the current configuration when the button *OK* is pressed. To edit a value double click on the respective entry.

Group definitions and color assignments are dragged and dropped from one trace file branch to another or even onto the branch *File Default* to make them available for each new trace file. The dialog box allows dragging and dropping every child node in the configuration tree, and thus the configuration is easily manipulated in a way that leads to surprising results up to and including crashes of the Intel® Trace Analyzer. In such cases, it should be started again with a well-known configuration.

Generally, it is best 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 hence, 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, color definitions can typically be dragged and dropped from trace to trace or to default (the latter would 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. Examination and possible value editing can solve such issues.



Option	Value
✓ File //knsfiler1/PSP/mosel/itatp/examples/poisson_icomm.single.stf	
✓ File Default	
✓ File //knsfiler1/home/gbisseli/traces/geofem/geofem_8x4x4.stf.single	
✓ File //knsfiler1/PSP/mosel/itatp/examples/poisson_sendrecv.single.stf	
✓ File poisson_icomm.single.stf	

Configuration File: C:/Documents and Settings/ssreekum\.itarc

Figure 5-13. The Edit Configuration Dialog

5.10.2. The Load Configuration Dialog

This dialog box allows to read configuration data from formerly saved files back into the program. The information to be loaded is chosen on a file per file basis.

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

5.11. The Find Dialog

The *Find* dialog box searches for a process/function. This dialog box is found in the context menu of the Function Profile and in the context menus of the Function Group Editors and the Process Group Editors.

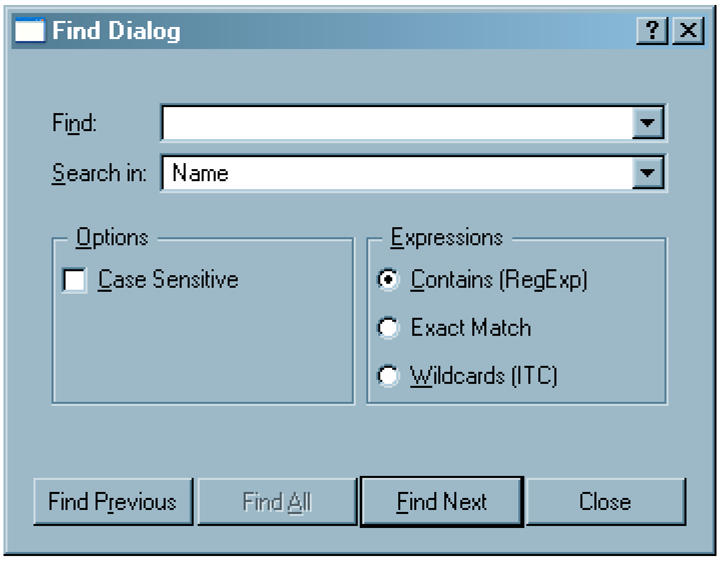


Figure 5-14. The Find dialog

Searching is made case-sensitive using the given check box in the *Find Dialog* box. A pull-down menu provides the option of searching by *Name*, by *All Columns* or by *Dynamic Path*.

The *Find* dialog box provides a few options with regard to the type of expression that optimize the search process. These are:

- *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. More on the valid regular expressions is found at http://doc.trolltech.com/3.3/qregexp.html#details.

- *Exact Match*

  This entry searches for the exact match of a given character or set of characters.

- *Wild Cards (Intel® Trace Collector)*

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 wild card characters in use are *, **, ? and []. These match any number of characters except for the colon(:). Pattern matching is case-insensitive.

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)

Chapter 6. Comparison

The most simple way to compare two trace files or two time intervals from the same trace file is to just open two Views and to show them next to each other. While this provides a rough overview a CVomparison View allows to calculate the exact differences and speedups between two runs or between two ranges of the same run. To open a Comparison View for two files just open them and choose **View Menu**⟶**View**⟶**Compare** in one of the Views. Choose the other file (which can be the same) from the dialog. A new Comparison View will be opened.

The rules that a View imposes on its Charts, namely that it enforces the same time interval, aggregation and filters on the Charts are extended for Comparison Views. The Comparison View holds two sets of time interval, aggregation and filters, one for each file. They are shown in the Views status area at the bottom just like in normal Views.

All Charts that were described in Chapter 4 sofar can only show data from a single trace file. This will stay the same and they will be tied exactly to one set of the constraints. If you choose **View Menu**⟶**Charts**⟶**Event Timeline** in a comparison View then actually two timelines will appear, one for each file.

Until now a Comparison View does not provide a real advantage over just having two regular Views. The real benefit comes in when you open one of the now available Comparison Charts. The Charts menu of a Comparison Chart contains new comparings variants of the profiles that will calculate differences and speedups between the two runs. These new Chart variants are explained in Section 6.2.

There is an additional menu entry **View Menu**⟶**Comparison** that provides some control over the Comparison Views behavior.

View Menu⟶**Comparison**⟶**Same Time Scaling** is switched off per default. If checked all timelines use the same scaling. That means that a timeline showing a time interval of one second appears exactly half as wide as another timeline that shows a time interval of two seconds. If time intervals differ by more than a factor of hundred, this setting is ignored and the timelines are aligned as usually.

View Menu⟶**Comparison**⟶**Couple Navigation Keys** is switched off per default. This switch controls the behavior of the navigation keys in comparison mode. When switched off the result of pressing a navigation key such as the right arrow depends on the Chart that currently has the focus. If the Chart belongs for example to file A then the time interval for A will change. If this switch is on then both time intervals will change by the same relative amount.

View Menu⟶**Comparison**⟶**Couple Mouse Zoom** is switched off per default. If switched on zooming with the mouse in a timeline that belongs to file A will zoom to a corresponding time interval in file B.

6.1. Mappings in Comparison Views

The proverbial error in doing any comparison is to compare apples and oranges. When comparing two program runs this could be 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 without recognizing 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 can not be determined automatically. There might be even no such mapping.

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 too impossible to even enumerate all combinations of parameters that might have changed between two runs and that were hold constant. To foresee all these cases in terms of a automagically adapting GUI is outright impossible.

The design of the Comparison View is based on the assumption that a scalability analysis of two runs with differening processor counts is the most frequent use case for comparison. This case should be supported without excluding any other use cases.

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

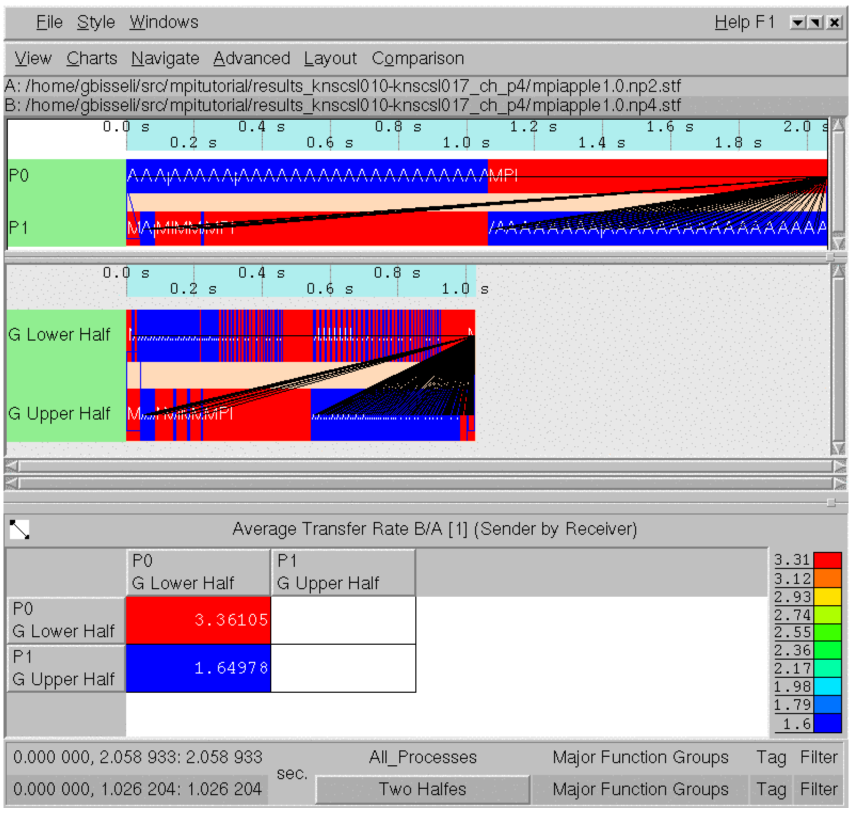
Communicators are mapped by their Ids. Message tags are mapped literally by their value.

The mapping of processes and process groups is controlled by choosing the process aggregations for both files as outlined in Section 6.1.1.

The mapping between functions and function groups is handled as outlined in Section 6.1.2.

6.1.1. 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 Comparison Message Profile that is meaningful under these assumptions choose the aggregation as shown in Figure 6-1. Here for the run B a process aggregations into two halfes was chosen.


File Style Windows
Help F1
View Charts Navigate Advanced Layout Comparison
0.0 s 0.2 s 0.4 s 0.6 s 0.8 s 1.0 s 1.2 s 1.4 s 1.6 s 1.8 s 2.0 s
P0
P1
MPI
0.0 s 0.2 s 0.4 s 0.6 s 0.8 s 1.0 s
G Lower Half
G Upper Half
MPI

A: /home/gbisseli/src/mpitutorial/results_knscsl010-knscsl017_ch_p4/mpiapple1.0.np2.stf
B: /home/gbisseli/src/mpitutorial/results_knscsl010-knscsl017_ch_p4/mpiapple1.0.np4.stf

Average Transfer Rate B/A [1] (Sender by Receiver)

	P0 G Lower Half	P1 G Upper Half
P0 G Lower Half	3.36105	
P1 G Upper Half	1.64978	

3.31	🟥
3.12	🟧
2.93	🟨
2.74	🟩
2.55	🟩
2.36	🟩
2.17	🟩
1.98	🟦
1.79	🟦
1.6	🟦

0.000 000, 2.058 933: 2.058 933 sec. All_Processes Major Function Groups Tag Filter
0.000 000, 1.026 204: 1.026 204 sec. Two Halfes Major Function Groups Tag Filter

Figure 6-1. Comparing run A with 2 processes to run B with 4 processes

The message profile shows the quotient B/A of the average transfer rate. The rule that the Comparison Message Profile uses to map the senders and receivers of the two runs onto each other is quite simple: child number *i* of run A's process aggregation is always compared with (mapped to) child number *i* of run B's process aggregation.

6.1.2. Mapping of Functions

...

6.2. Comparison Charts

6.2.1. The Comparison Function Profile

...

6.2.2. The Comparison Message Profile

...

6.2.3. The Comparison Collective Operations Profile

...

Chapter 7. Concepts

This Chapter contains explanations of some abstract concepts of Intel® Trace Analyzer.

7.1. 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.

The 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.

The 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 it should be stressed 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.

7.2. Aggregation

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

7.2.1. 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 Intel® Trace Analyzer selecting Aggregation into the predefined process group *All_Nodes* 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 that selecting a process group generally results in displaying the information for the group's children (with the notable exception of the function profile). That is the reason why single, unthreaded processes or single threads cannot be selected for aggregation.

Until now, there was only a hierarchy with two levels, but more complicated hierarchies are useful too: 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 etc. These considerations suggest allowing for deeply nested thread groups.

If the thread group representing a single node is selected to concentrate on intra-node effects then the analysis might appear to be not faster but slower than using the thread group *All_Processes* alone. Why is that? The reason is twofold. First the Intel® Trace Analyzer doesn't have to do any aggregation for the thread group *All_Processes* since it is flat (assuming no threads are used). Second, 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 **Views Menu**⟶**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 Intel® Trace Analyzer.

7.2.2. Function Aggregation

Aggregation into function groups allows 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 Section 7.2.1 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. This gets clearer if the binary instrumentation feature of Intel® Trace Collector is used as the result is thousands of functions instrumented.

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 on precisely the code that is considered tunable while code that is controlled by third parties is aggregated into coarse categories.

Note that selecting a function group generally results in displaying the information for the groups' children. That is the reason why single functions cannot be selected for aggregation.

7.3. Tagging and Filtering

Both concepts use the same filter grammar in their filter expressions. See Section 5.1 and Section 5.2 for usage hints.

Conceptually there is a different filter for each class of events: function events, messages and collective operations. During analysis the right filter expression is evaluated for each event and the event is tagged or passed on, if the expression is true while it is left untagged or suppressed, if the expression is false.

7.3.1. Tagging

If several events are merged as described in Section 7.1 then the merged event is tagged if at least one of the singular events is tagged.

Therefore tagging in particular together with the Qualitative Timeline Chart (see Section 4.2) is a very powerful tool to find events matching a certain criterion. This is because only if a single event out of billions matches the criteria of the tag filter, then it is guaranteed that there will be a highlighted peak in the Qualitative Timeline Chart that indicates where to zoom into the trace.

7.3.2. 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 that lets all functions except MPI pass, then there won't be holes in the Event Timeline, instead it will look as if MPI was not called. This means that it will look 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.

What happens if there are functions A, B and C in the trace where A calls B and B calls C and then suppress B by filtering it out? It would appear as if A had called C directly! This is quite different from choosing a function aggregation that does not cover B, because that will show the function group *other* wherever B was shown before. Again: Filtering and Aggregation are orthogonal.