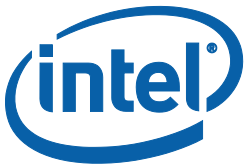


Tutorial: Reducing Trace File Size

Intel® Trace Analyzer and Collector

Contents

Legal Information

1. Overview

Intel® Trace Analyzer and Collector enables you to understand MPI application behavior and quickly find bottlenecks to achieve high performance for parallel cluster applications. Intel Trace Collector generates trace files of MPI applications, while Intel Trace Analyzer visualizes the MPI application behavior using the generated trace file.

Large trace files are hard to manage: it takes a long time to generate them, a lot of disk space to store them, and they are difficult to analyze with the Intel® Trace Analyzer. For these reasons, you need reduce the trace file size whenever possible. Learn how to reduce trace file size with the Intel® Trace Collector.

About This Tutorial	This tutorial demonstrates a workflow applied to the `poisson` application. You can ultimately apply the same steps to your own application(s). The options to reduce the trace file size are: • Create configuration file • Run the application with the `-trace-collectives` and `-trace-pt2pt` options • Use the `MPI_Pcontrol` function After you reduced the trace file size, generate the trace file.
Estimated Duration	10-15 minutes.
Learning Objectives	After you complete this tutorial, you should be able to: • Reduce trace file size using various Intel® Trace Collector functionality • Generate trace files of reduced size
More Resources	Learn more about the Intel® Trace Analyzer and Collector in the User and Reference Guides available at: • *Intel® Trace Analyzer User and Reference Guide* • *Intel® Trace Collector User and Reference Guide* The guides are available at Intel Trace Analyzer and Collector Product Page.

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

Key Terms

Configuration File

1.1. Prerequisites

This section describes the steps you need to do before you start using the Intel® Trace Analyzer and Collector.

1.1.1. Required Software

To perform all the steps described in this tutorial, you will need the following software installed on your system:

- Intel® compilers

- Intel® MPI Library
- Intel® Trace Analyzer and Collector

All of these products are installed as part of Intel® Parallel Studio XE Cluster Edition.

1.1.2. Setting Up the Environment Variables

Linux* OS:

Set the required environment variables by sourcing the `psxevars.c[sh]` script available at `<install-dir>/parallel_studio_xe_<version>.x.xxx/bin`, where `<install-dir>` is the Intel® Parallel Studio XE Cluster Edition installation directory. For example:

```
$ source psxevars.sh
```

Windows* OS:

Open the command prompt from **Start > Intel Parallel Studio XE *version* > Compiler and Performance Libraries > Intel 64 Visual Studio *version* environment**. This will set all required environment variables and you will be ready to trace your applications.

1.1.3. Creating Trace Files

To trace the `poisson` application, go to `<install-dir>/examples/poisson` and copy its contents into your working directory, then trace the application:

Linux* OS:

1. Compile the application running the `make` command. Adjust the `Makefile`, if necessary.
2. Run the application with the `-trace` option of `mpirun`:

   ```
   $ mpirun -n 4 -trace ./poisson
   ```

Windows* OS:

1. Compile all components of the `poisson` application from the compiler command prompt with the `-trace` option. The basic command line for Fortran programs is:

   ```
   > mpiifort -trace myApp.f90
   ```

2. Run the application to generate a tracefile:

   ```
   > mpiexec -n 4 myApp.exe
   ```

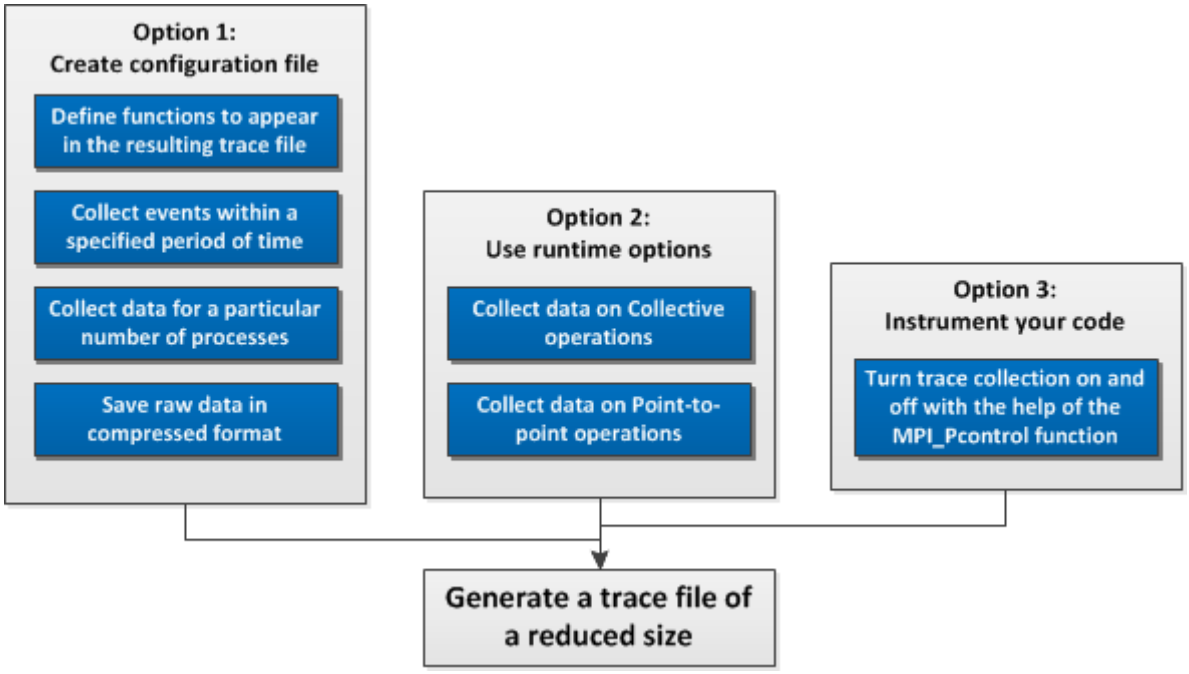
For your convenience, this tutorial comes with a set of trace files available at `<install-dir>/examples/traces`.

Once you have the trace data available for the `poisson` application, you are ready to start the analysis.

2. Reducing Trace File Size

Use Intel® Trace Collector for MPI applications to generate trace files of reduced size that you can later analyze in the Intel® Trace Analyzer tool and finally improve the application performance.

This tutorial demonstrates how to reduce the size of the trace file you are going to collect on the example of the `poisson` sample code.



Option 1: Create Configuration File	Define the trace data filters in the Intel® Trace Collector configuration file.
Option 2: Use Runtime Options	Set the Intel® Trace Collector to collect data on Collective or Point-to-Point operations.
Option 3: Instrument Your Code	Include the `MPI_Pcontrol` function call into the source code to generate a trace file for a particular segment.

Key Terms

Configuration File

2.1. Create Configuration File

Create a new text file in a text editor and save it with the `.conf` extension. Add the necessary filtering options into the configuration file to make the Intel® Trace Collector trace only the data you want to appear in the resulting trace file. To make Intel Trace Collector read the file, set its full path in the `VT_CONFIG` environment variable. See instructions for each filtering setting below.

Use the following filter settings:

- STATE defines the functions that will appear in the collected trace file. STATE filters functions by their full names, for example: `MPI:MPI_Send`.
- ACTIVITY is a shortcut for STATE. It filters functions by their class names, for example: `MPI`.

- SYMBOL is a shortcut for STATE. It filters functions by their names, excluding class name, for example: `MPI_Send`.
- PROCESS enables trace collection for a particular number of processes.
- TIME-WINDOWS collects events within the specified period of time.
- COMPRESS-RAW-DATA saves raw data in the compressed format.

To get detailed information about filtering process, filtering options, and configuration syntax, refer to the *Intel® Trace Collector User and Reference Guide*.

Key Terms

Configuration File

2.1.1. STATE

Specify the STATE option in the configuration file to define the function that matches the pattern.

To do this:

1. Create a new configuration file.
2. In the configuration file, specify STATE to collect the trace data for the `MPI_Allreduce` and `MPI_Sendrecv` functions:

   ```
   STATE MPI:* OFF
   STATE MPI:*Allreduce* ON
   STATE MPI:*Sendrecv* ON
   ```

3. Save the configuration file: `my_settings_state.conf`.
4. Set the `VT_CONFIG` environment variable to the directory that contains the configuration file:

   ```
   VT_CONFIG=/<configuration file path name>/my_settings_state.conf
   ```

5. Run the application.

   Linux* OS:

   ```
   $ mpirun -trace -n 4 ./poisson
   ```

   Windows* OS:

   ```
   > mpiexec -n 4 poisson.exe
   ```

Key Terms

Configuration File

2.1.2. ACTIVITY

The ACTIVITY option is a shortcut for STATE `"<pattern>:*"`.

To exclude the MPI calls from the output trace file, set the ACTIVITY filter to OFF.

To do this:

1. Create a new configuration file.
2. In the configuration file, specify the ACTIVITY option:

   ```
   ACTIVITY MPI OFF
   ```

3. Save the configuration file: `my_settings_activity.conf`.
4. Set the `VT_CONFIG` environment variable to the directory that contains the configuration file:

   ```
   VT_CONFIG=/<configuration file path name>/my_settings_activity.conf
   ```

5. Run the application.

Linux* OS:

```
$ mpirun -trace -n 4 ./poisson
```

Windows* OS:

```
> mpiexec -n 4 poisson.exe
```

Key Terms

Configuration File

2.1.3. SYMBOL

The SYMBOL option is a shortcut for STATE "**:*<pattern>*".

Define the SYMBOL setting in the configuration file if you want the Intel® Trace Collector to collect data on all `Allreduce` functions, both from MPI and from user code.

To do this:

1. Create a new configuration file.
2. In the configuration file, specify the SYMBOL option:

   ```
   SYMBOL *:Allreduce ON
   ```

3. Save the configuration file: `my_settings_symbol.conf`.
4. Set the VT_CONFIG environment variable to the directory that contains the configuration file:

   ```
   VT_CONFIG=/<configuration file path name>/my_settings_symbol.conf
   ```

5. Run the application.

   Linux* OS:

   ```
   $ mpirun -trace -n 4 ./poisson
   ```

   Windows* OS:

   ```
   > mpiexec -n 4 poisson.exe
   ```

Key Terms

Configuration File

2.1.4. PROCESS

Enable trace collection for a specific number of processes using the PROCESS option.

To do this:

1. Create a new configuration file.
2. To get the trace data for four processes, specify the PROCESS option in the configuration file:

   ```
   PROCESS 0:N OFF
   PROCESS 0:3 ON
   ```

3. Save the configuration file. Specify the name of the file, for example, `my_settings_process.conf`.
4. Set the VT_CONFIG environment variable to the directory that contains the configuration file:

   ```
   VT_CONFIG=/<configuration file path name>/my_settings_process.conf
   ```

5. Run the application.

   Linux* OS:

   ```
   $ mpirun -trace -n 4 ./poisson
   ```

Windows* OS:

```
> mpiexec -n 4 poisson.exe
```

2.1.5. Generate a Trace File

To check how the configuration file influences the trace collection process, run four processes of the `poisson` sample code with the following configuration file:

1. Create a new configuration file.
2. Type in the following parameters:

   ```
   LOGFILE-FORMAT STFSINGLE
   ACTIVITY MPI OFF
   STATE MPI:*Allreduce* ON
   ```

3. Save the configuration file, for example, `filter.conf`.
4. Set the `VT_CONFIG` environment variable to the directory that contains the configuration file:

   ```
   VT_CONFIG=/<path_name>/filter.conf
   ```

5. Run the application.

   Linux* OS:

   ```
   $ mpirun -trace -n 4 ./poisson
   ```

   Windows* OS:

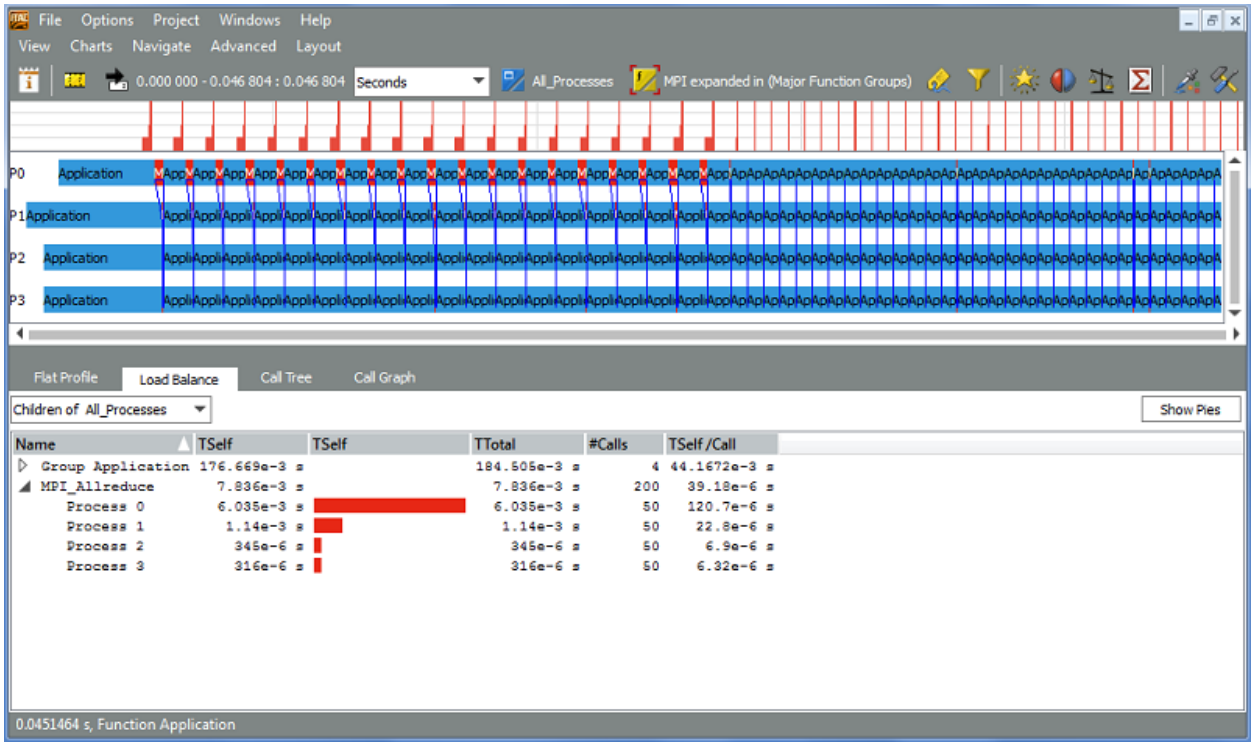
   ```
   > mpiexec -n 4 poisson.exe
   ```

To check the results, do the following:

Check your work:

1. Open the resulting trace file `poisson.single.stf` in Intel® Trace Analyzer.
2. In the Function Profile, right-click **Group MPI** and select **Ungroup Group MPI** from the context menu.
3. Go to the **Load Balance** tab of the Function Profile to see how the `MPI_Allreduce` function is distributed among the four processes.
4. Go to **Charts > Event Timeline** to see the activities in each of the four processes individually.

The result looks like:



You can see that the trace data was collected only for the `MPI_Allreduce` function.

Key Terms

Configuration File
Event Timeline
Function Profile

2.1.6. TIME-WINDOWS

Use the `TIME-WINDOWS` option to set up time frames within which Intel® Trace Collector will save the events into the trace file.

You can have several time frames at the same run.

To set time windows for Intel Trace Collector to collect trace data for the `poisson` application from `0` to 28 milliseconds only, do the following:

1. Create a new configuration file.
2. To get the correct order of messages in the output trace file, include the first communication into the first time window.
3. For the `poisson` application, set up the time frame from `0` to 28 milliseconds using the following syntax (the lower-case "L" character stands for milliseconds):

   ```
   TIME-WINDOWS 0:28l
   ```

   For more information about the syntax of parameters, refer to the *Intel® Trace Collector User and Reference Guide*.
4. Save the configuration file as `my_settings_timewindows.conf`.
5. Set the `VT_CONFIG` environment variable to the directory that contains the configuration file:

   ```
   VT_CONFIG=/<configuration file path name>/my_settings_timewindows.conf
   ```
6. Run the application.

   Linux* OS:

   ```
   $ mpirun -trace -n 4 ./poisson
   ```

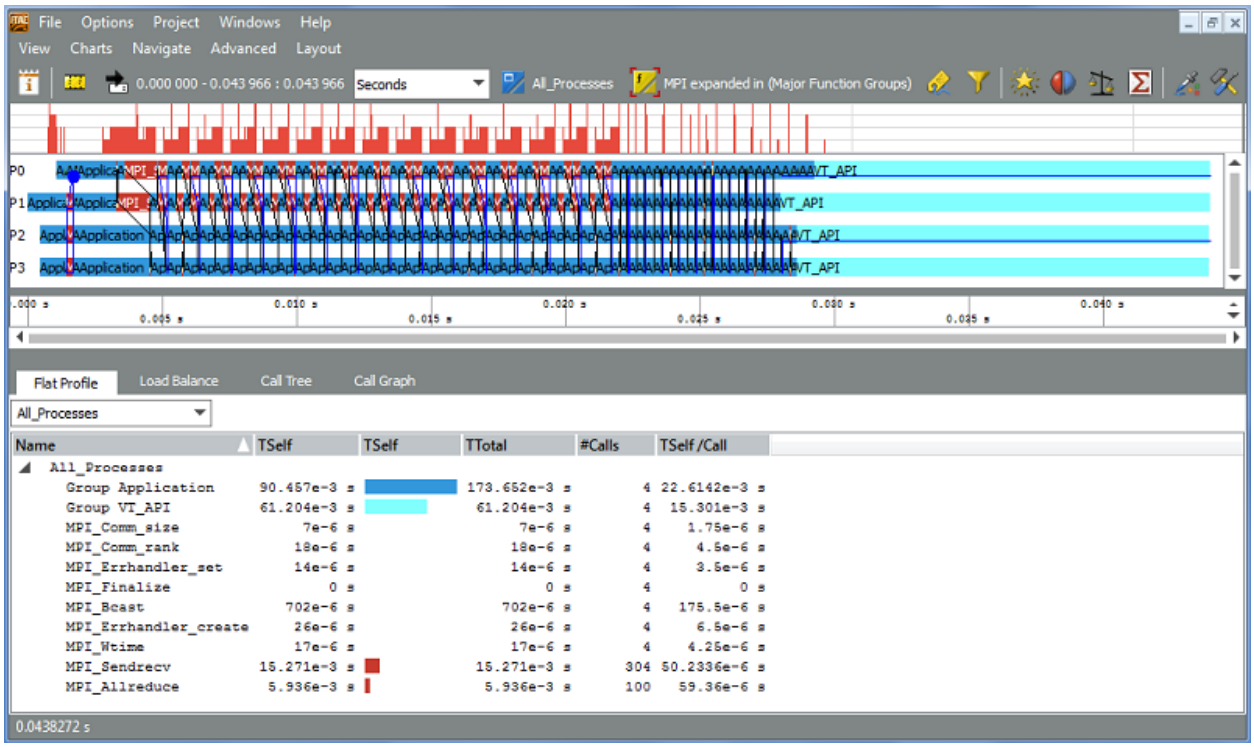
Windows* OS:

```
> mpiexec -n 4 poisson.exe
```

Check the resulting trace file:

1. Open the resulting trace file `poisson.stf` in the Intel® Trace Analyzer.
2. In the Function Profile, right-click **Group MPI** and select **Ungroup Group MPI** from the context menu.
3. Go to **Charts > Event Timeline** to see the activities in each of the four processes individually.
4. Go to **Charts > Time Scale** or press the toolbar button to open the time scale.

You can see that the total execution time has been written into the trace file, but the trace data is collected only for the specified period of time (28 milliseconds). The `VT_API` group stands for the part of the code where trace data collection was off:



Key Terms

Configuration File
Event Timeline
Function Profile

2.1.7. COMPRESS-RAW-DATA

Use the `COMPRESS_RAW_DATA` option to make Intel® Trace Collector store raw data in compressed format.

The compression works for raw events only. The merge process will be performed either by the `stftool` utility or by Intel® Trace Analyzer when you open the collected trace.

To do this:

1. In a new configuration file, set the `KEEP_RAW_EVENTS` and `COMPRESS_RAW_DATA` to `ON`:

   ```
   KEEP_RAW_EVENTS ON
   COMPRESS_RAW_DATA ON
   ```

2. Save the configuration file as `my_settings_compress.conf`.

3. Set the VT_CONFIG environment variable to the directory that contains the configuration file:

```
VT_CONFIG=/<configuration file path name>/my_settings_compress.conf
```

4. Run the application.

   Linux* OS:

```
$ mpirun -trace -n 4 ./poisson
```

   Windows* OS:

```
> mpiexec -n 4 poisson.exe
```

Key Terms

Configuration File

2.2. Use Runtime Options

Use the `-trace-collectives` and `-trace-pt2pt` options to get information only about the Collective and Point-to-Point operations in the application.

- Get Trace Data on Collective Operations
- Get Trace Data on Point-to-Point Operations

NOTE

These options are only supported at runtime with the Hydra Process Manager.

NOTE

The `–trace-collectives` and `–trace-pt2pt` options use the configuration file specially set for these options and redefine the VT_CONFIG environment variable. When you use these options, Intel® Trace Collector will not read your own configuration file.

2.2.1. Get Trace Data on Collective Operations

To collect information about the Collective operations in the `poisson` application, run the application using the commands below.

Linux* OS:

```
$ mpirun –trace –trace-collectives –n 4 ./poisson
```

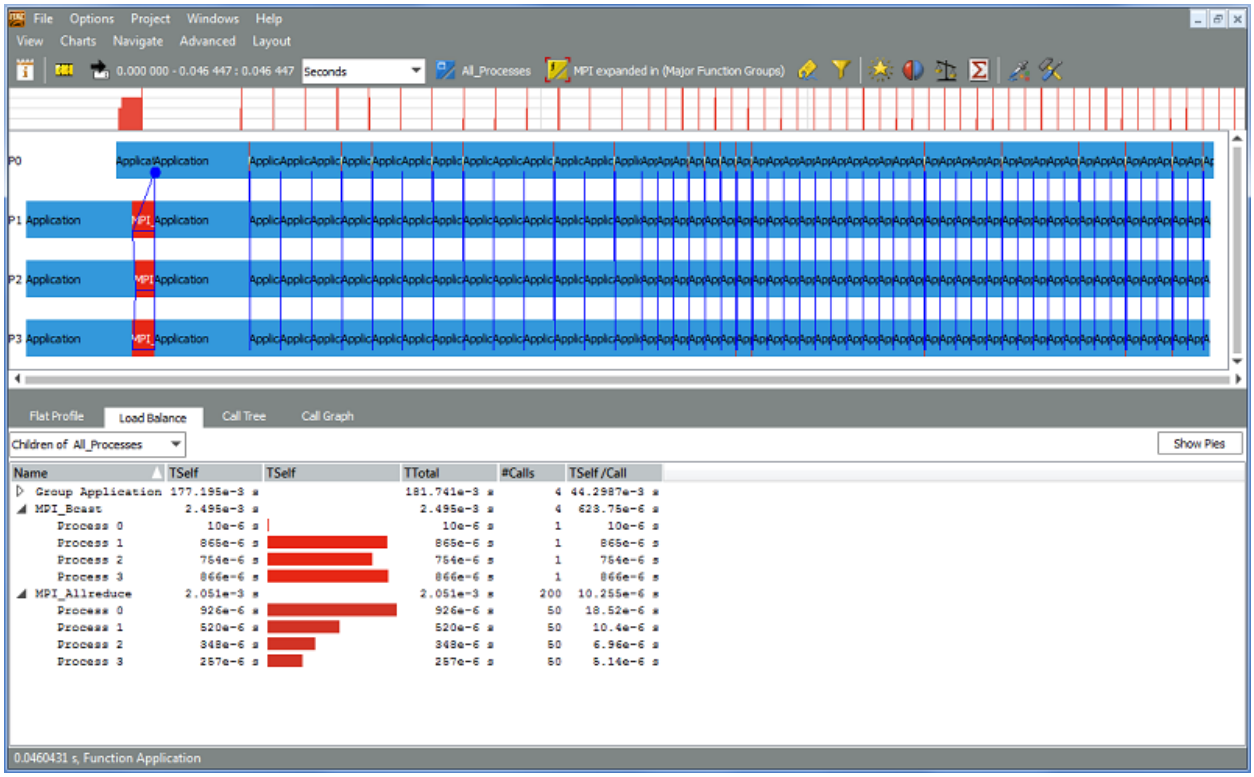
Windows* OS:

```
> mpiexec –trace-collectives –n 4 poisson.exe
```

Check your work:

1. Open the resulting trace file `poisson.stf` in Intel® Trace Analyzer.
2. In the Function Profile, right-click **Group MPI** and select **Ungroup Group MPI** from the context menu.
3. Go to the **Load Balance** tab of the Function Profile to see how the `MPI_Bcast` and `MPI_Allreduce` collectives are distributed among the four processes.
4. Go to **Charts > Event Timeline** to see the activities in each of the four processes individually.

You can see that the trace data was collected only for the Collective operations that are presented by two functions: `MPI_Bcast` and `MPI_Allreduce`:



Key Terms

Configuration File
Event Timeline
Function Profile

2.2.2. Get Trace Data on Point-to-Point Operations

To collect data on the Point-to-Point operations in the `poisson` application, run the application using the command:

Linux* OS:

```
$ mpirun –trace -trace-pt2pt –n 4 ./poisson
```

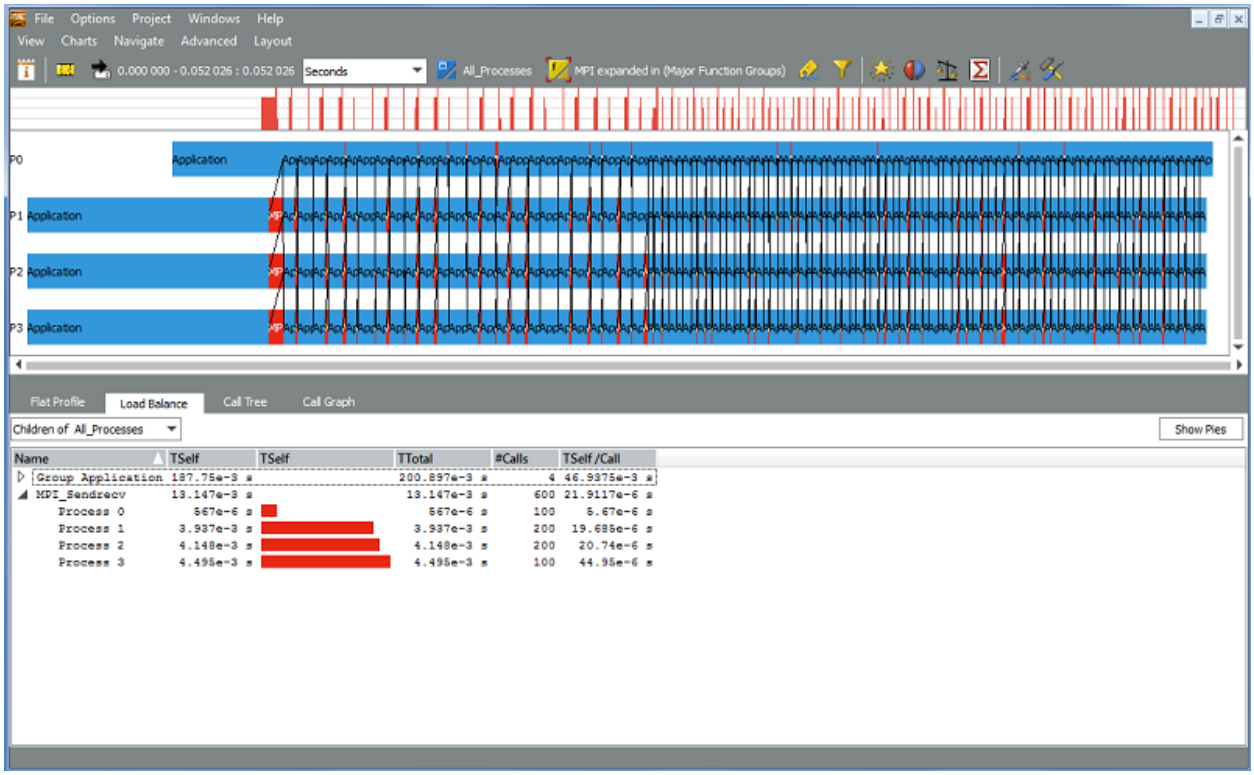
Windows* OS:

```
> mpiexec -trace-pt2pt –n 4 poisson.exe
```

Check your work:

1. Open the resulting trace file `poisson.stf` in the Intel® Trace Analyzer.
2. In the Function Profile, right-click **Group MPI** and select **Ungroup MPI** from the context menu.
3. Go to the **Load Balance** tab of the Function Profile to see how the `MPI_Sendrecv` function is distributed among the four processes.
4. Go to **Charts > Event Timeline** to see the activities in each of the four processes individually.

You can see that the trace data was collected only for the Point to Point operations that are presented by the `MPI_Sendrecv` function:



Key Terms

Configuration File
Event Timeline
Function Profile

2.3. Instrument Your Code

Insert the `MPI_Pcontrol` function call into your application to turn the trace collection on and off.

To collect the data on the `exchange` function of the `poisson` application, do the following:

1. Go to `<installdir>/examples/poisson/pardat.f90`
2. Edit the `pardat.f90` file:
    1. Right after the `MPI_Init` call, turn the trace collection off:
    ```
    MPI_Pcontrol(0);
    ```
    2. At the beginning of the `poisson_red_black` subroutine, turn the trace collection on:
    ```
    MPI_Pcontrol(1);
    ```
    3. At the end of the `poisson_red_black` subroutine, turn the trace collection off
3. Save the `pardat.f90` file.
4. Rebuild the application.
5. Run the application.

    Linux* OS:

    ```
    $ mpirun -trace -n 4 ./poisson
    ```

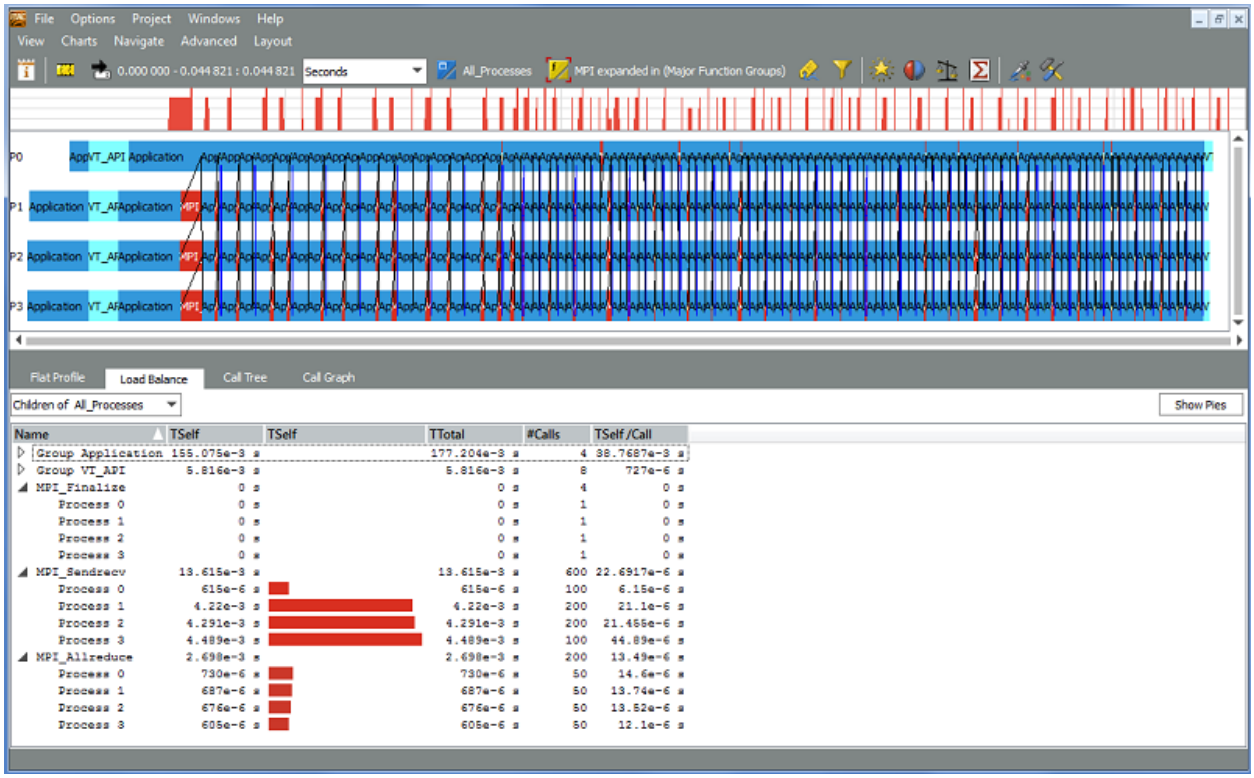
    Windows* OS:

    ```
    > mpiexec -n 4 poisson.exe
    ```

Check your work:

1. Open the resulting trace file `poisson.stf` in Intel® Trace Analyzer.
2. In the Function Profile, right-click **Group MPI** and select **Ungroup Group MPI** from the context menu.
3. Go to the **Load Balance** tab of the Function Profile to see how the `MPI_Finalize`, `MPI_Sendrecv` and `MPI_Allreduce` functions are distributed among the four processes.
4. Go to **Charts > Event Timeline** to see the activities in each of the four processes individually.

You can see that the trace data was collected only for a particular part of the code. Note the VT_API group: it stands for the part of the code where trace data collection was turned off:



Key Terms

Configuration File
Event Timeline
Function Profile

3. Summary

You have completed the *Reducing Trace File Size* tutorial. The following is the summary of the important things to remember when using the Intel® Trace Collector capabilities for reducing the size of your trace file.

Option	Tutorial Recap	Key Tutorial Take-aways
1. Configuration options	• Set the `STATE` option to collect data on the `MPI_Allreduce` and `MPI_Sendrecv` function calls. • Set `SYMBOL` to collect data on `Allreduce` function calls. • Set `ACTIVITY` to collect data on all MPI calls. • Set `PROCESS` to collect data for four processes: from Process 0 to Process 3 • Set time frames. • Set raw data compression.	You can define several filters in one and the same configuration file.
2. Runtime options	• Ran the application with the `-trace-collectives` option to get information only on Collective operations. • Used the runtime option `-trace-pt2pt` to get data on Point-to-Point operations in the application.	`-trace-collectives` and `-trace-pt2pt` options are only supported at runtime with the Hydra process manager.
3. Instrument your code to collect data on specific functions	Included the `MPI_Pcontrol` function call into the source code to collect trace data on the `exchange` function.	Total execution time will be written into the trace file, but the trace file will contain only the MPI functions called in the part of code indicated by `MPI_Pcontrol`.

Next step: Apply one of the described methods to reduce the size of your own trace file.

4. Key Terms

The following terms are used throughout this tutorial:

Configuration file: A file that contains a set of configuration parameters for the Intel® Trace Collector. It is a plain ASCII file containing a number of directives, one per line. Each directive consists of an identifier followed by arguments.

Event timeline: A chart that displays individual process activities over time. Horizontal bars represent the processes with the functions called in these processes. The bars consist of colored rectangles labeled with the function names. Black lines indicate messages sent between processes. These lines connect sending and receiving processes. Blue lines represent collective operations, such as broadcast or reduce operations.

Function profile: A chart that provides detailed profiling information on the performance data. The Function Profile consists of four tabs: **Flat Profile**, **Load Balance**, **Call Tree** and **Call Graph**. These tabs use the same column headers and the same raw data.