

Tracing the Execution of MPI Applications with Windows HPC Server 2008 R2 with Service Pack 3 (SP3)

Published: May 10, 2012

**Abstract**

You can pair the version of Microsoft® MPI (MS-MPI) available for Windows® HPC Server 2008 R2 starting with Service Pack 3 (SP3) with the Event Tracing for Windows (ETW) infrastructure in the Windows client and server operating systems to trace MPI applications for performance analysis and troubleshooting. This article describes how to use ETW in combination with Windows HPC Server 2008 R2 starting with SP3 to trace MPI applications.

# Copyright Information

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2012 Microsoft. All rights reserved.

Contents

[Summary 4](#_Toc323764724)

[Tracing an MS-MPI Application 6](#_Toc323764725)

[Sample Scripts to Automate Synchronizing and Formatting 10](#_Toc323764726)

[Script to Synchronize and Convert to Text 11](#_Toc323764727)

[Script to Synchronize and Convert to OTF 12](#_Toc323764730)

[Script to Synchronize and Convert to CLOG 13](#_Toc323764733)

[Understanding Trace Output 15](#_Toc323764736)

[Reading Text Event Logs 15](#_Toc323764737)

[Viewing CLOG2 Trace Files with Jumpshot 18](#_Toc323764738)

[The Pieces You Need 18](#_Toc323764739)

[Using Jumpshot – Step-by-Step 19](#_Toc323764740)

[Jumpshot Features 20](#_Toc323764741)

[Advanced Tracing Topics 24](#_Toc323764742)

[Troubleshooting Tips 24](#_Toc323764743)

[Mpicsync Errors with the Message, “…a rank appears more than once…” 24](#_Toc323764744)

[Tapping into live ETW Traces 25](#_Toc323764745)

[Appendix 1: MS-MPI Event Filters 26](#_Toc323764746)

[Appendix 2: Related Information 27](#_Toc323764747)

# Summary

You can pair the version of Microsoft® MPI (MS-MPI) available for Windows® HPC Server 2008 R2 starting with Service Pack 3 (SP3) with the Event Tracing for Windows (ETW) infrastructure in the Windows client and server operating systems to trace MPI applications. You can trace MPI applications for the following purposes:

* **Performance Analysis** — Synchronized, high-precision timing data enables precise analysis of where an application spends execution time.
* **Application Troubleshooting** — The entry and exit of each MPI API call is logged, including the data that was passed in and out of the function.

The combination of MS-MPI and ETW enables you to perform the following tasks:

* **Create traces in production environments.**

You do not need special builds to create traces. You can run your MPI application with the **mpiexec** command. Then, if you have the Windows Performance Toolkit installed, you can run the **xperf -start** and **xperf -stop** commands when you expect your MPI application to reach points when you want to start and strop tracing. These **xperf** commands generate trace files that you can use for subsequent analysis.

* **Create MPI event** **logs from all processes on all nodes** **running an MPI application.**
* Trace any MS-MPI API call, including the ability to specify that you want to trace events generated from any of several built-in API groups.
* Trace interconnect-level activity for insight into lower level communication within the socket, shared memory, and Network Direct implementations of MS-MPI.
* Customize the trace generation to create trace logs from the operating system, driver, MPI, and applications (requires an ETW-enabled driver and application).
* **Synchronize event logs by using high-precision CPU clock correction for MS-MPI.**

To make sense of events from applications that run across many compute nodes and communicate in a few millionths of a second, you need ultra-high-precision correlation of the computer clocks on those nodes. A tool in Windows HPC Server 2008 R2, called MPI Clock Synchronization (**mpicsync**) accounts for the clock differences and the drift over time of those differences. The **mpicsync** command reads, computes the time correction, and alters the time-stamps for each event in the trace files for a job. The **mpicsync** command assesses the clock drift based upon the communications performed by the application without actually interfering with those communications or adding additional MPI calls, thereby reducing the effect of tracing on the run-time behavior of the application.

* **Format trace data for display in a number of viewers.**

You can format the event logs in the following formats:

* Text format with high resolution timestamps
* Open Trace Format (OTF)
* CLOG2 format (event-based logging format from Argonne National Labs)

You can use the following tools to view the trace data:

* Jumpshot (Argonne National Labs visualization tool)

<http://www-unix.mcs.anl.gov/perfvis/software/viewers/index.htm>

* Visual Studio® & Windows® ETW tools (not MPI-specific)

<http://msdn.microsoft.com/en-us/library/bb385774.aspx>

<http://msdn.microsoft.com/en-us/library/bb968803(VS.85).aspx>

* Vampir Viewer for Windows (Open Trace Format (OTF) visualization tool)
* **Tap into live event streams.**

Another useful feature of ETW is the ability for applications to tap into live event streams. The ETW infrastructure is described more fully on the Microsoft Developer Network (MSDN) website (<http://msdn.microsoft.com/en-us/library/bb968803(VS.85).aspx>).

Tracing Requirements

Keep in mind the following requirements as you plan tracing activities on your Windows HPC Server 2008 R2 with SP3 cluster:

* You must run trace jobs with a user account in either the Administrator or *Perform*ance Log Users group. This requirement is a security measure of the Event Tracing for Windows subsystem. If you use the Performance Log Users group, the cluster administrator must add this group to the Users group on the head node, from which Windows HPC Server 2008 R2 with SP3 replicates the group members to all of the compute nodes.
* You should run all trace jobs as Exclusive in Windows HPC Server 2008 R2 with SP3. By running the tracing job as Exclusive, you ensure that a single job will be running on the compute nodes for the duration of each trace job, thereby avoiding confusion and conflicts in the trace data.

# Tracing an MS-MPI Application

You can use commands in Windows HPC Server 2008 R2 starting with SP3 and in the Windows Performance Toolkit to create event traces for you MS-MPI application. In this example, you run your MPI application by using the **mpiexec** command, and then use the **xperf** command from the Windows Performance Toolkit to start and stop tracing when you expect the MPI application to perform operations of interest.

For information about installing the Windows Performance Toolkit, see [Installation](http://msdn.microsoft.com/en-us/library/ff190927(VS.85).aspx) (http://msdn.microsoft.com/en-us/library/ff190927(VS.85).aspx).

For an illustration of these steps, see [Figure 1](#Figure1).

1. **Use the mpiexec command to start your MS-MPI application.**

You use the [**mpiexec**](http://technet.microsoft.com/en-us/library/cc947675(WS.10).aspx) command to run an MS-MPI application on an HPC cluster. The following example runs the MyApplication.exe application with *N* parameters:

mpiexec MyApplication.exe param1 param2 … paramN

You typically run the **mpiexec** command by specifying it as the command line for a task in an HPC job. For example:

job submit /numnodes:<number\_of\_nodes> /requestednodes:<node\_list> mpiexec MyApplication.exe param1 param2 … paramN

**Note**  The **-trace** and **-tracefile** parameters for the **mpiexec** command are deprecated in Windows HPC Server 2008 R2 starting with SP3. If you use these parameters to trace an MS-MPI application for the entire time that it runs, you can generate large log files that can be difficult to analyze and can fill up the available disk space on the nodes that run the application. You should manually start and stop tracing instead to get information for a portion of the application that is of interest.

1. **Start tracing by running the xperf -start command.**

If you have the Windows Performance Toolkit installed, you can start tracing by using the [**xperf ‑start**](http://msdn.microsoft.com/en-us/library/ff191015(VS.85).aspx) command. To run the **xperf -start** command on multiple nodes, use the [**clusrun**](http://technet.microsoft.com/en-us/library/cc947685(WS.10).aspx) command to run the **xperf -start** command. For example:

clusrun –nodes:node1,node2[,...] xperf –start mpi –on Microsoft-HPC-MPI[:*hexadecimal\_flag\_value*]

You can specify that you want the trace to include information about a subset of events, such as those events that a specific group of APIs generate, by specifying one of the hexadecimal values from the table in [Appendix 1](#_Appendix_1:_MS-MPI) as part of the value for the **-on** parameter. To include all MPI-related events in your trace files, omit the hexadecimal value.

1. **Stop tracing by running the xperf -stop command.**

If you have the Windows Performance Toolkit installed, you can start tracing by using the [**xperf ‑stop**](http://msdn.microsoft.com/en-us/library/ff191018(VS.85).aspx) command. To run the **xperf -stop** command on multiple nodes, use the **clusrun** command to run the **xperf -stop** command. For example:

clusrun –nodes:node1,node2[,...] xperf –stop mpi -d C:\TracingFiles\MyApplication\_trace.etl

**Important** You should always explicitly stop tracing by running the **xperf -stop** command. Tracing does not stop automatically when your MPI application or the job that runs the MPI application finishes running. So, if you do not explicitly stop tracing, tracing continues when the MPI application and job are finished, and the event logs can continue to grow until the compute nodes run out of disk space. If you need further control of the size of the event logs, you can use the **-MaxFile** option of the **xperf -start** command to specify the maximum size of the log file. For example, if you run an MS-MPI application on a virtual machine (VM) role in Windows Azure that has a size of Extra Large, the virtual hard disk for the node has a size of 10 gigabytes (GB). In that case, you should use the **-MaxFile** option to set the maximum size of the log file to smaller value, such as 1 GB.

**Note**  The events from all processes on a given compute node are written to a single trace file on the compute node. This behavior results in trace log files being written on each node on which the job is running. By default, this trace file is saved at C:\User.etl. Use the **-d** option of the **xperf -stop** command to save the trace data to a file with a more meaningful name. The information in the trace file for each event includes the ID of the physical processor core, operating system process IDs, and so on, which you can use to group events in your trace visualization tool of choice.

By default, two ring buffers of 20 GB each are used when tracing MS-MPI applications, one buffer for communication events, and one for API events. You can change these defaults by using the [**wevtutil**](http://technet.microsoft.com/en-us/library/cc732848(WS.10).aspx) command. For example, to increase the size of the log files to 60 GB each and set the log retention mode so that new incoming events overwrite the older events when the file reaches its maximum size, run the following commands:

wevtutil sl Microsoft-HPC-MPI/Communication /ms:62914560 /rt:false

wevtutil sl Microsoft-HPC-MPI/Api /ms:62914560 /rt:false

1. **Create the CPU clock synchronization data for each process.**

Use the [**mpicsync**](http://technet.microsoft.com/en-us/library/cc947673(WS.10).aspx) command line tool to correct the trace file timestamps for each node used in a job that ran an MPI application. The **mpicsync** command is an MPI program, so you use the **mpiexec** command to run **mpicsync** and quickly calculate the clock corrections simultaneously on the nodes used to generate the trace. For example:

mpiexec –cores 1 mpicsync C:\TracingFiles\MyApplication\_trace.etl

**Note**  The **mpicsync** command solely uses trace (.etl) file data to calculate CPU clock corrections. You can synchronize trace files immediately after a run, or at a later time–and on the same or different computers–with no loss of clock correction accuracy. The trace data from all processes on a compute node are written to a single trace file (one for each job) on each node, so you must run a single **mpicsync** process on each compute node. You can accomplish running a single **mpicsync** process on each compute node by using the **-cores 1** option of the **mpiexec** command.

1. **Format the binary .etl file for viewing and analysis.**

You can convert the .etl files to text, Open Trace Format (OTF), or CLOG2 files.

* Convert the trace log to text format.

Use the [**tracerpt**](http://technet.microsoft.com/en-us/library/cc732700(WS.10).aspx) command to convert the event log to a text file. For example:

mpiexec -cores 1 tracerpt C:\TracingFiles\MyApplication\_trace.etl -rts -o C:\TracingFiles\MyApplication\_trace.txt -of CSV

* Convert the trace logs to OTF.

Use the Windows HPC Server 2008 OTF translator, etl2otf.exe, to format the event log as an OTF file. The OTF translatoris an MPI program, so you use the **mpiexec** command to run [**etl2otf**](http://technet.microsoft.com/en-us/library/cc972772(WS.10).aspx) and spread the formatting work across all of the nodes used in the job. For example:

mpiexec -cores 1 etl2otf C:\TracingFiles\MyApplication\_trace.etl

The **etl2otf** command constructs a base output file name by appending \_otf to the trace (.etl) file name, and then creates the output files listed in the following table in Open Trace Format:

|  |  |  |
| --- | --- | --- |
| **File Name** | **File Name Example** | **Description** |
| ***etl\_file\_name*\_otf.otf** | mpi\_trace.etl\_otf.otf | OTF master file |
| ***etl\_file\_name*\_otf.0.def** | mpi\_trace.etl\_otf.0.def | OTF global definitions file |
| ***etl\_file\_name*\_otf.*n*.events** | mpi\_trace.etl\_otf.1.events  mpi\_trace.etl\_otf.2.events  ... | Event files, one for each MPI rank. |

* Convert the trace logs to CLOG v.2 format for the Jumpshot viewer.

Use the Windows HPC Server 2008 CLOG translator, etl2clog.exe, to format the event log as a CLOG event file for use with the Jumpshot viewer from Argonne National Lab. The CLOG converteris an MPI program, so you use the **mpiexec** command to run [**etl2clog**](http://technet.microsoft.com/en-us/library/cc972868(WS.10).aspx) and spread the formatting work across all of the nodes used in the job and collect the results into a single output (CLOG) file on the rank 0 node. For example:

mpiexec -cores 1 etl2clog C:\TracingFiles\MyApplication\_trace.etl

The **etl2clog** command constructs an output file name by appending .clog2 to the trace (.etl) file name as the translator creates CLOG v. 2 formatted files. Unlike **tracerpt** and **etl2otf,** which create local output files on each compute node that you must be collect, **etl2clog** gathers this information itself and creates a single output file with a .clog2 extension.

1. **Copy the formatted event files to a single location and, optionally, merge them into a single, time-correlated log of MPI events on all processes and all nodes.**
2. Use standard operating system commands to copy the files to a file share on the head node, and append the compute node computer name to the file name to prevent the files from being overwritten in the copy operation. For example:

set TRACE\_DIR=\\%CCP\_SCHEDULER%\TracingShare\%USERNAME%

mkdir %TRACE\_DIR%\Text

mpiexec -cores 1 cmd /V:ON /C copy /y C:\TracingFiles\MyApplication\_trace.txt "**%TRACE\_DIR%\Text**\MyApplication\_trace\_!COMPUTERNAME!.txt"

**Note**  The above example illustrates moving a text file to a head node file share, but you can use a similar approach for OTF files as well. The example uses the **/V:ON** option of the **cmd** command and encloses the COMPUTERNAME environment variable in exclamation marks (!) to delay the environment variable expansion so that expansion takes place on each compute node instead of just the lead compute node which received the mpiexec command.

1. Use standard operating system commands to merge and sort the text-based event files on the head node file share. For example:

copy /y ["%TRACE\_DIR%\Text\MyApplication\_trace\*.txt](file:///\\headnode\%25USERNAME%25\%25CCP_JOBID%25\mpi_trace*.txt)" ["%TRACE\_DIR%\Text\MyApplication\_combined\_trace\_unsorted.txt](file:///\\headnode\%25USERNAME%25\%25CCP_JOBID%25\mpi_trace.etl.unsorted.txt)"

sort ["%TRACE\_DIR%\Text\MyApplication\_combined\_trace\_unsorted.txt](file:///\\headnode\%25USERNAME%25\%25CCP_JOBID%25\mpi_trace.etl.unsorted.txt)" /o ["%TRACE\_DIR%\Text\MyApplication\_combined\_trace\_sorted.txt](file:///\\headnode\%25USERNAME%25\%25CCP_JOBID%25\mpi_trace.etl.sorted.txt)"

Figure 1 : MS-MPI Trace Process



# Sample Scripts to Automate Synchronizing and Formatting

These scripts enable you to synchronize and format the trace files for your application by running a single-task job that runs one command. Separate scripts are provided for each of the following formats:

* Text
* OTF
* CLOG

**Installation**: To run one of these scripts on any set of nodes in your cluster, the script that you want to run must be accessible to all of the compute nodes. Therefore, you should copy the script to a cluster file share or to the local hard drive of each node.

In these examples, the scripts are placed in a shared folder on the head node named TracingShare.

## Script to Synchronize and Convert to Text

The SyncConvertToText example synchronizes the .etl files from tracing an MPI application and formats those trace files as a text file that contains comma-separated values (CSV) that provide information about the events. You can run the example on an HPC cluster by creating a single-task job.

### SyncConvertToText Usage

job submit /numnodes:<number\_of\_nodes\_used\_for\_trace> /requestednodes:<node\_list> \\%CCP\_SCHEDULER%\TracingShare\SyncConvertToText.cmd <trace\_file\_path\_and\_name>.etl

### SyncConvertToText Source Code

@rem -------------------------- SyncConvertToText.cmd --------------------------

@echo off

@rem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IMPORTANT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@rem SyncConvertToText assumes that the cluster administrator created a

@rem shared folder on the head node named TracingShare for the use of

@rem developers on the cluster. The output files are copied to this share

@rem from all compute nodes at the end of the script.

@rem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@rem

@rem Set the root folder for the user on the file share on the head node.

@rem

set TRACE\_DIR=\\%CCP\_Scheduler%\TracingShare\%USERNAME%

@rem

@rem Set the name of the Event Tracing for Windows (.etl) file

@rem from the input to the script.

@rem

set ETL\_FILE=%1

set ETL\_FILE\_NO\_PATH=%~n1%~x1

@rem

@rem Synchronize the clocks, using just one process on each compute node.

@rem

mpiexec -cores 1 mpicsync %ETL\_FILE%

@rem

@rem Format the trace file as text.

mpiexec -cores 1 tracerpt %ETL\_FILE% -rts -o %ETL\_FILE%.txt -of CSV

@rem

@rem Copy the text files.

@rem

mkdir %TRACE\_DIR%\Text

mpiexec -cores 1 cmd /V:ON /C copy /y %ETL\_FILE%.txt "%TRACE\_DIR%\Text\%ETL\_FILE\_NO\_PATH%\_!COMPUTERNAME!.txt"

@rem

@rem Merge and sort the text files.

@rem

copy /y "%TRACE\_DIR%\Text\%ETL\_FILE\_NO\_PATH%\*.txt" "%TRACE\_DIR%\Text\%ETL\_FILE\_NO\_PATH%\_combined\_unsorted.txt"

sort "%TRACE\_DIR%\Text\%ETL\_FILE\_NO\_PATH%\_combined\_unsorted.txt" /o "%TRACE\_DIR%\Text\%ETL\_FILE\_NO\_PATH%\_combined\_sorted.txt"

@rem Remove the trace files from the compute nodes.

@rem

@rem OPTION: Add "@rem" to the beginning of the following line to

@rem retain the trace files on each compute node after

@rem running a trace job.

mpiexec -cores 1 cmd /c del %ETL\_FILE%\*

## Script to Synchronize and Convert to OTF

The SyncConvertToOTF example synchronizes the .etl files from tracing an MPI application and formats those trace files as OTF files. You can run the example on an HPC cluster by creating a single-task job.

### SyncConvertToOTF Usage

job submit /numnodes:<number\_of\_nodes\_used\_for\_trace> /requestednodes:<node\_list> \\%CCP\_SCHEDULER%\TracingShare\SyncConvertToOTF.cmd <trace\_file\_path\_and\_name>.etl

### SyncConvertToOTF Source Code

@rem -------------------------- SyncConvertToOTF.cmd --------------------------

@echo off

@rem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IMPORTANT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@rem SyncConvertToOTF assumes the cluster admin has created a shared folder

@rem on the head node named TracingShare for the use of developers on

@rem the cluster. The output files are copied to this share from all

@rem compute nodes at the end of the script.

@rem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@rem

@rem Set the root folder for the user on the file share on the head node.

@rem

set TRACE\_DIR=\\%CCP\_Scheduler%\TracingShare\%USERNAME%

@rem

@rem Set the name of the Event Tracing for Windows (.etl) file and the nodes

@rem on which the application was traced from the input to the script.

@rem

set ETL\_FILE=%1

set ETL\_FILE\_NO\_PATH=%~n1%~x1

@rem

@rem Synchronize the clocks, using just one process on each compute node.

@rem

mpiexec -cores 1 mpicsync %ETL\_FILE%

@rem

@rem Format the trace file as OTF.

@rem

mpiexec -cores 1 etl2otf %ETL\_FILE%

@rem

@rem Copy the OTF files.

@rem

mkdir %TRACE\_DIR%\OTF

mpiexec -cores 1 cmd /C if exist %ETL\_FILE%\_otf.otf copy /y %ETL\_FILE%\_otf.otf "%TRACE\_DIR%\OTF\%ETL\_FILE\_NO\_PATH%\_otf.otf"

mpiexec -cores 1 cmd /C if exist %ETL\_FILE%\_otf.0.def copy /y %ETL\_FILE%\_otf.0.def "%TRACE\_DIR%\OTF\%ETL\_FILE\_NO\_PATH%\_otf.0.def"

mpiexec -cores 1 cmd /C copy /y %ETL\_FILE%\_otf.\*.events "%TRACE\_DIR%\OTF"

@rem Remove the trace files from the compute nodes.

@rem

@rem OPTION: Add "@rem" to the beginning of the following line to

@rem retain the trace files on each compute node after

@rem running a trace job.

mpiexec -cores 1 cmd /c del %ETL\_FILE%\*

## Script to Synchronize and Convert to CLOG

The SyncConvertToCLOG example synchronizes the .etl files from tracing an MPI application and formats those trace files as a CLOG file. You can run the example on an HPC cluster by creating a single-task job.

### SyncConvertToCLOG Usage

job submit /numnodes:<number\_of\_nodes\_used\_for\_trace> /requestednodes:<node\_list> \\%CCP\_SCHEDULER%\TracingShare\SyncConvertToCLOG.cmd <trace\_file\_path\_and\_name>.etl

### SyncConvertToCLOG Source Code

@rem -------------------------- SyncConvertToCLOG.cmd --------------------------

@echo off

@rem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IMPORTANT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@rem SyncConvertToCLOG assumes the cluster admin has created a shared folder

@rem on the head node named TracingShare for the use of developers on

@rem the cluster. The output files are copied to this share from all

@rem compute nodes at the end of the script.

@rem \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@rem

@rem Set the root folder for the user on the file share on the head node.

@rem

set TRACE\_DIR=\\%CCP\_Scheduler%\TracingShare\%USERNAME%

@rem

@rem Set the name of the Event Tracing for Windows (.etl) file and the nodes

@rem on which the application was traced from the input to the script.

@rem

set ETL\_FILE=%1

set ETL\_FILE\_NO\_PATH=%~n1%~x1

@rem

@rem Synchronize the clocks, using just one process on each compute node.

@rem

mpiexec -cores 1 mpicsync %ETL\_FILE%

@rem

@rem Format the trace file as a CLOG2 file.

@rem

mpiexec -cores 1 etl2clog %ETL\_FILE%

@rem

@rem Copy the CLOG2 file.

@rem

mkdir %TRACE\_DIR%\CLOG

mpiexec -cores 1 cmd /C if exist %ETL\_FILE%.clog2 copy /y %ETL\_FILE%.clog2 "%TRACE\_DIR%\CLOG\%ETL\_FILE\_NO\_PATH%.clog2"

@rem Remove the trace files from the compute nodes.

@rem

@rem OPTION: Add "@rem" to the beginning of the following line to

@rem retain the trace files on each compute node after

@rem running a trace job.

mpiexec -cores 1 cmd /c del %ETL\_FILE%\*

# Understanding Trace Output

## Reading Text Event Logs

The text output file that the **tracerpt** command creates contains a line for each event that occurred, plus a line of column headings at the start of the file. Each line consists of several columns that contain information about the event. The following table describes the columns in the text version of the event log file.

|  |  |
| --- | --- |
| Column | Description |
| EventName | The name of the event. For MS-MPI events, this name is HPC MPI Event Provider. |
| Type | The type of the event. The following table shows the possible values that appear in this column and the opcodes to which those values correspond:   |  |  | | --- | --- | | Type | Opcodes | | 0 | 0 (Win:Info) | | Start | 1 (Win:Start) | | Stop | 2 (Win:Stop) | | queu | 10 (net:queue) | | qcon | 11 (net:connect) | | head | 12 (net:head) | | inln | 13 (net:inline) | | cont | 14 (net:continue) | | done | 15 (net:done), 16 (net:packet), 17 (net:data) | |
| EventID | The numeric identifier for the event. |
| Version | The version of the event. Different versions of an event can include different sets of data. |
| Channel | The channel for the event. A channel is used to collect events. For MPI events, this value is either 16 (Windows HPC MPI Communication Event Channel) or 17 (Windows HPC MPI API Event Channel). |
| Level | The severity level for the event, which indicates the severity or the verbosity of the event. The level can be one of the following values:   |  |  | | --- | --- | | Level | Description | | 1 | Critical | | 2 | Error | | 3 | Warning | | 4 | Informational | | 5 | Verbose | |
| Opcode | The operation code for the event, which identifies the activity or a point within an activity that the application was performing when it raised the event. For a list of operation codes for MPI events, see the <opcodes> section of the %CCP\_HOME%\bin\Mpitrace.man manifest file for your installation of Microsoft HPC Pack 2008 R2. |
| Task | The task for the event, which identifies a major component of the event provider. For a list of operation codes for MPI events, see the <tasks> section of the %CCP\_HOME%\bin\Mpitrace.man manifest file for your installation of Microsoft HPC Pack 2008 R2. |
| Keyword | The keyword used to classify the event. For MPI events, this keyword is one of the hexadecimal values from the table in Appendix 1 that corresponds to a group name that begins with an mpi prefix. |
| PID | The identifier of the Windows process that generated the event. |
| TID | The identifier of the Windows thread that generated the event. |
| Processor Number | The identifier of the Windows logical processor (core) on which the thread that generated the event was running. |
| Instance ID | Unused for MS-MPI events. |
| Parent Instance ID | Unused for MS-MPI events. |
| Activity ID | Unused for MS-MPI events. |
| Related Activity ID | Unused for MS-MPI events. |
| Clock-Time | The corrected clock time at which the event occurred. This time is synchronized across all compute nodes taking part in the tracing job, if you ran **mpicsync** on the .etl files.  If you used the **-rts** option when you ran **tracerpt** to create the text file, this value is a raw timestamp. If you did not use the **-rts** option when you ran **tracerpt**, this value is a file time. A file time is a 64-bit value that represents the number of 100-nanosecond intervals that have elapsed since 12:00 A.M. January 1, 1601 Coordinated Universal Time (UTC). |
| Kernel(ms) | The amount of processor time in kernel mode that the event used, in milliseconds. |
| User(ms) | The amount of processor time in user mode that the event used, in milliseconds. |
| User Data | For MS-MPI API events, this column is a notation of entry or exit (entry/leave) of an MS-MPI API function and the data entering/leaving the function – the argument values. For MS-MPI interconnect events, this column includes information describing the interconnect event, including the source rank, destination rank and message sequence. |

## Viewing CLOG2 Trace Files with Jumpshot

### The Pieces You Need

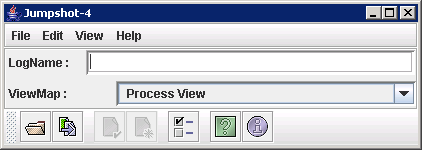
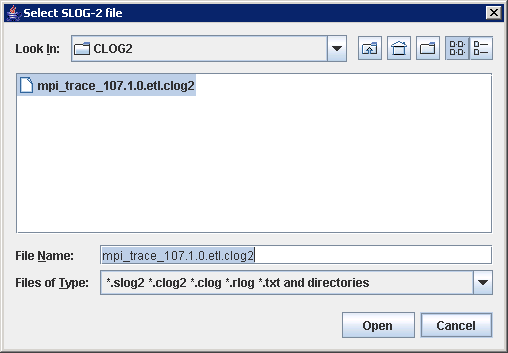
* **Online Jumpshot Manual from Argonne National Lab (ANL)**:  
   <http://www-unix.mcs.anl.gov/perfvis/software/viewers/jumpshot-4/usersguide.html>
* **Windows MPE Runtime Environment** (which includes Jumpshot):   
  MPI Parallel Environment (MPE) for Windows from Argonne National Lab: <ftp://ftp.mcs.anl.gov/pub/mpi/slog2/slog2rte.tar.gz>   
  And, optionally, some related downloads:   
  <http://www-unix.mcs.anl.gov/perfvis/download/index.htm>
* **A Java Virtual Machine of your choice**  
  Jumpshot is Java-based, so you need a JVM on the computer you use to view traces (not the whole Windows HPC Server 2008 R2 cluster). Per Argonne National Lab, Jumpshot is known to run in almost all Java2 runtimes. For your convenience, you can get Java Runtime Environment (JRE) for Windows Version 6 from Sun at <http://java.com/en/download/ie_manual.jsp?locale=en>.

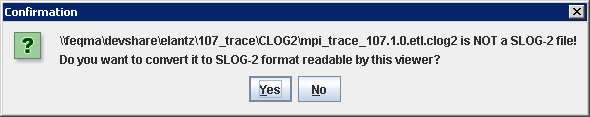
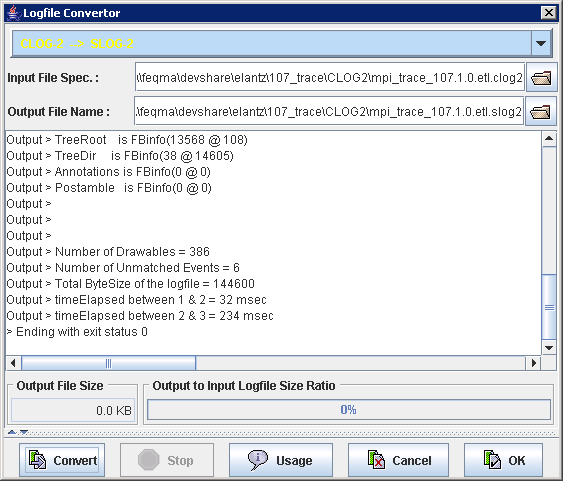
**Note**  You do not need the full SDK to use Jumpshot. The runtime environment is sufficient.

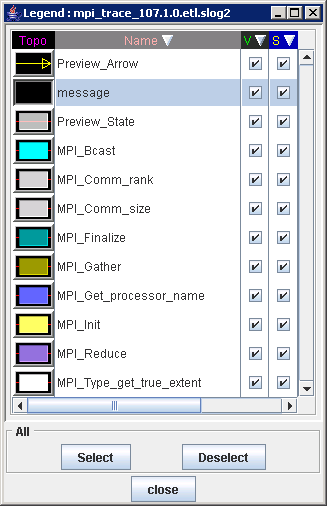
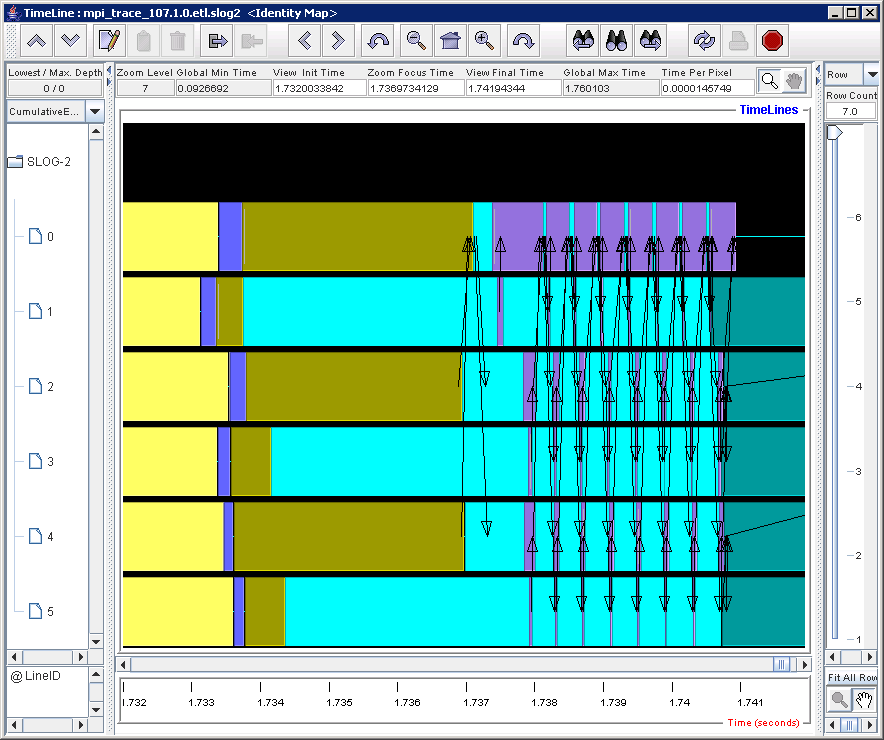
### Using Jumpshot – Step-by-Step

1. Install Software

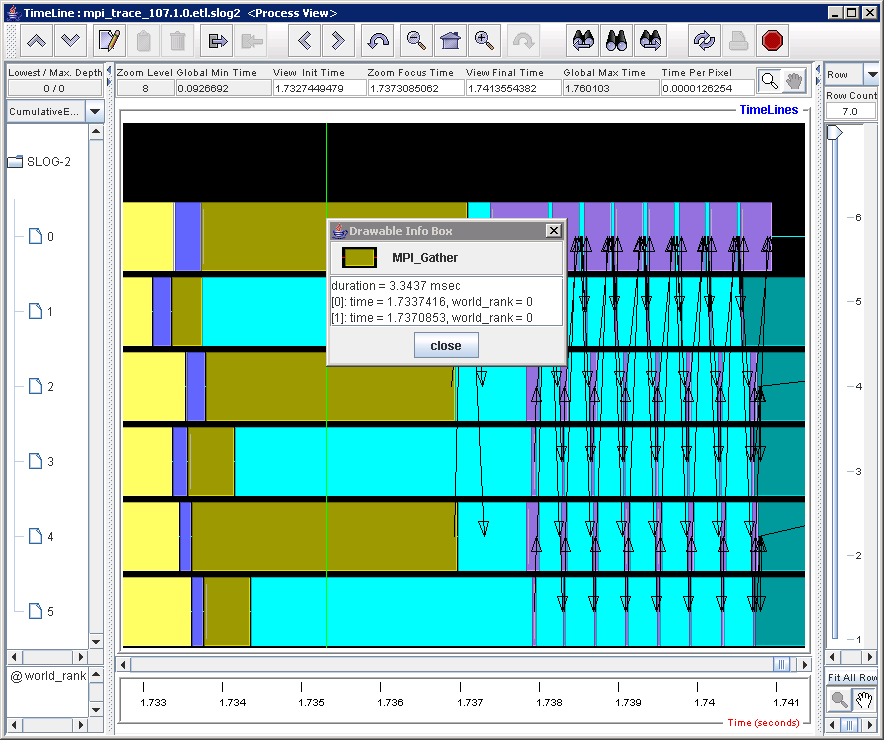
* Install a Java runtime environment on your development computer.
* Install the Windows MPE runtime environment on your development computer.

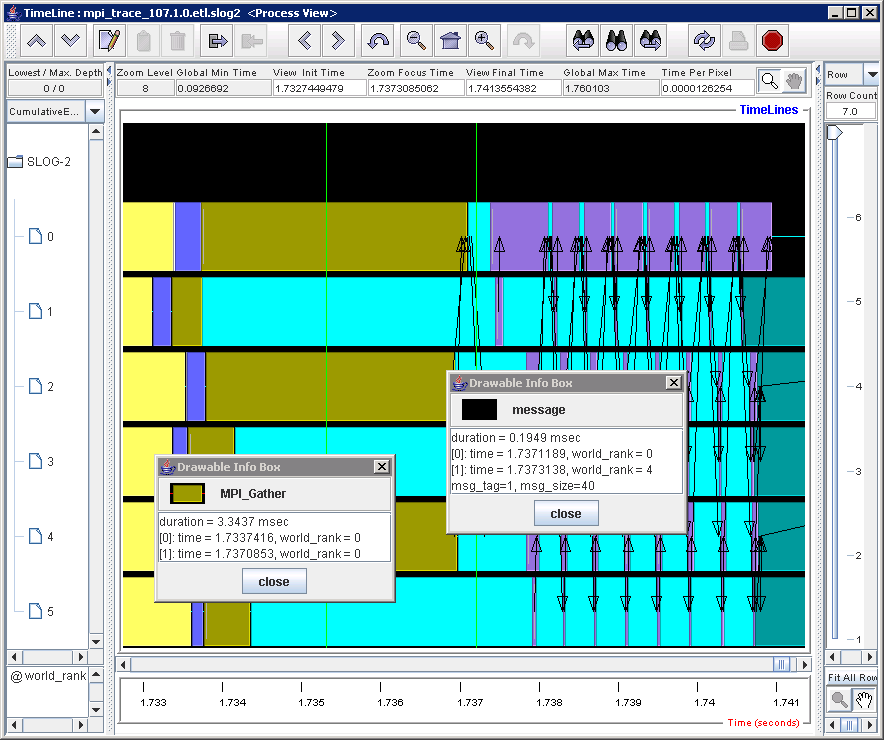
1. Create MS-MPI traces and use *etl2clog.exe* to convert them to CLOG2 format as described is some detail above.
2. Start Jumpshot  
   (adjust the paths to match your install locations, of course):   
   C:\>"C:\Program Files\Java\jre1.5.0\_09\bin\**java**" **-jar** "C:\slog2rte-install\lib\**jumpshot.jar**"
3. Choose File…Select from the pull-down menus and select your .clog2 file.

1. You will be prompted to convert your file. Click “Yes”, then “Convert” and finally “OK” because Jumpshot has to work from a SLOG2 file instead of the CLOG2 format generated by *etl2clog*.  Jumpshot will automatically save the SLOG2 conversion in the same folder as your CLOG2 file.
2. The Timeline window will be displayed next and now you’re ready to have some fun!



### Jumpshot Features

“Right-Click” on an Event to See More Detail

“Right-Click” on a Message Arrow to See More Detail

Note on physical versus logical representation of message exchange:   
You may notice, when viewing your MS-MPI traces in Jumpshot, some intermediate lines in your colored event rectangles and that message arrows are sometimes shown in unintuitive places. The MS-MPI traces track the actual physical message exchanges which took place during the trace job. This is in contrast with tools which depict logical message exchange where the message arrows start neatly at the beginning of send events and end at the conclusion of the corresponding receive event. While perhaps more complex to read, depiction of physical exchanges can provide enhanced insight into an application’s operation and was the choice of the Microsoft HPC team. There are currently no visualization applications available capable of depicting both message types.

Send Event

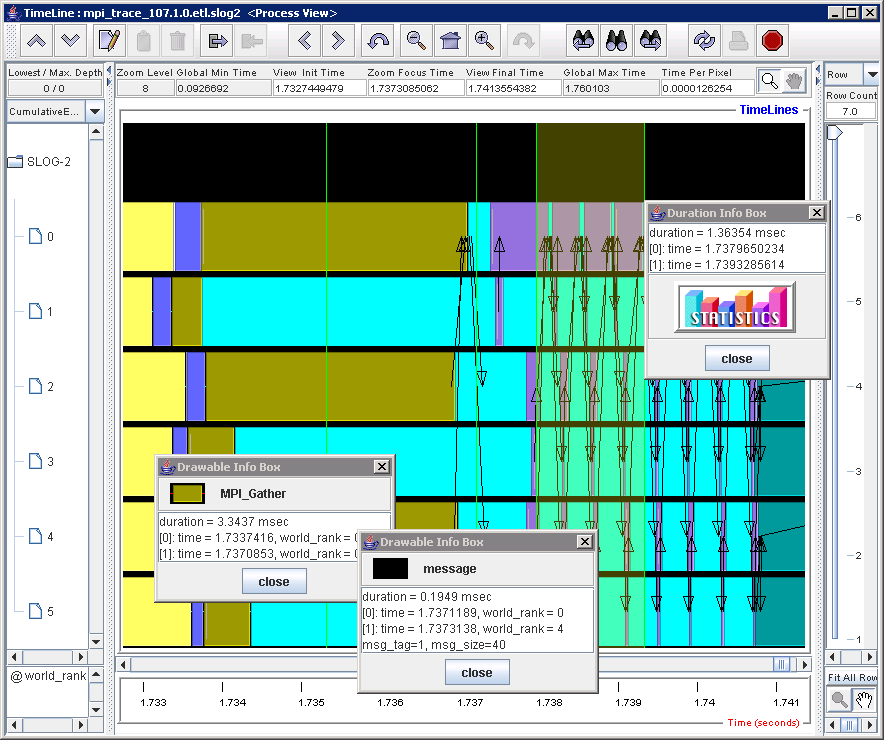
Receive Event

Physical Message Exchange

Logical Message Exchange

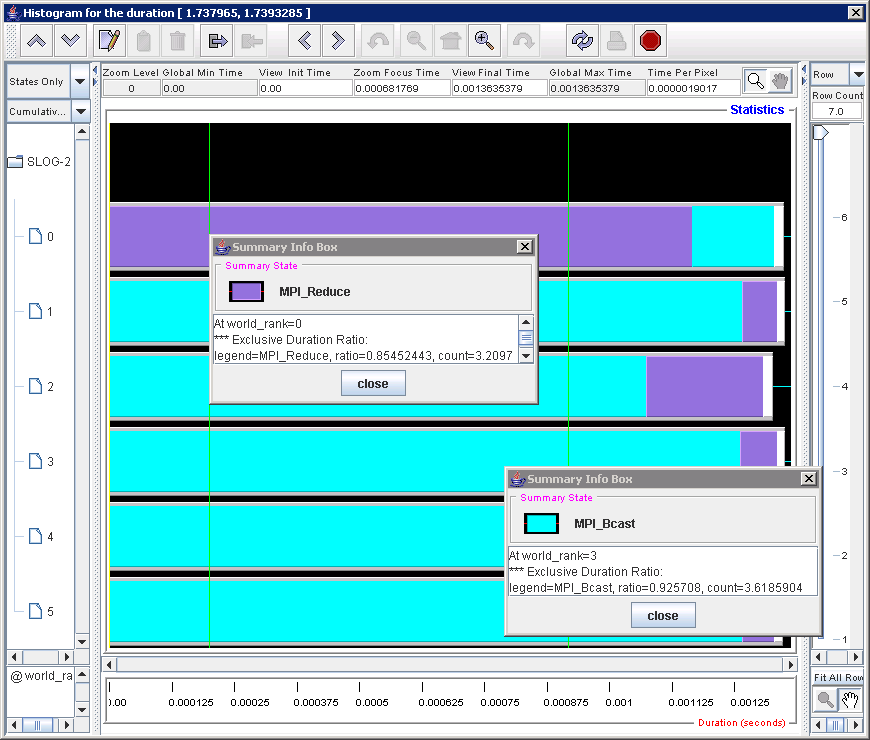
“Right-Click and Drag” to Highlight a Time Interval of Interest

Note the beginning and ending times in the Duration Info Box and a “Statistics” button



Highlighted Time Interval

Duration Info Box

The Statistics display shows the cumulative amount of time that each process spent on each type of MPI operation during the highlighted time interval. Note in the example below that rank 0 spent most of its communication time in MPI\_REDUCE whereas the other processes spent more communication time in MPI\_Bcast.

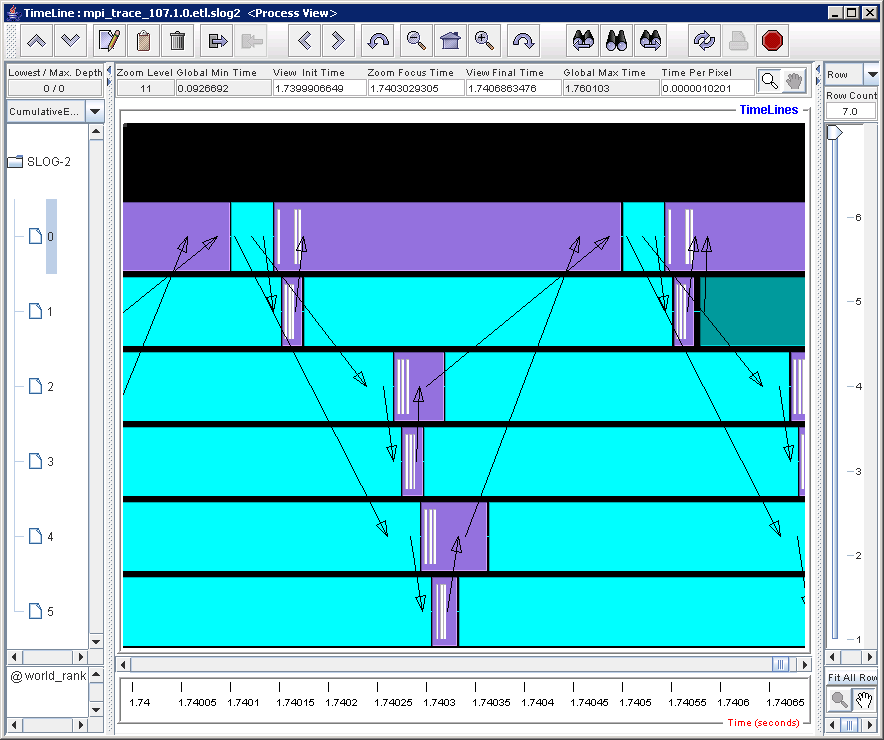
Learn…Optimize… (Repeat)

Developers can gain new levels of understanding with a visualization tool that are difficult to achieve any other way. For example, the trace snippet below shows a cycle of broadcasts from Rank 0 followed by reductions back to Rank 0. The C code is simply:

MPI\_Bcast(&NumIntervals, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

but if you follow the arrows (in the screenshot below) you’ll note the actual communication that occurs is Rank 0 to Ranks 1, 2, and 4. And then Rank2->Rank3 and Rank4->Rank5. This is no mistake as MS-MPI’s MPI\_Broadcast communicates via a binomial tree to increase efficiency. And you can see MPI\_Reduce works in a similar fashion. Is this something a developer can use to advantage? Maybe. The point is that though a visualization tool like Jumpshot, you can learn more about the operation of your code across the cluster and thus better optimize its operation.

This example illustrates the benefits of depicting the physical message exchange.



# Advanced Tracing Topics

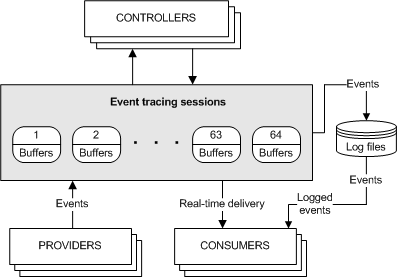
## Troubleshooting Tips

### Mpicsync Errors with the Message, “…a rank appears more than once…”

Be sure to run only one tracing job at a time on any given node. Otherwise the trace logs (.etl files) on those nodes can be overwritten, which causes inconsistencies with the trace logs on other nodes that were not overwritten.

## Tapping into live ETW Traces

When you use **xperf** or another tool to start and stop event tracing, that tool functions as an event controller. The API set of MS-MPI functions as an event provider that streams events into the session while the application runs. These events are written to a log file (.etl file) as in the illustration, but there is an ETW API for creating real-time event consumers that enables you to monitor an MPI application as it runs or build up a statistical representation of long-running applications for which a log file would be of cumbersome size. The following diagram shows the relationships among event tracing controllers, providers, consumers, and sessions.



For example, the Microsoft HPC team created a diagnostic tool to help optimize process placement for a long-running application according to the communication patterns between MPI ranks (processes). Because event log files can be very large and the Microsoft HPC team was looking for statistical data and not debugging a specific code path, the team chose to create a tool using the Windows ETW APIs that monitors the live event stream from all the nodes as the nodes run the application, The tool then builds up the statistical summarization of communication patterns. The tool discards event data as it builds the summary, so the tool does not have to content with or process any large files.

# Appendix 1: MS-MPI Event Filters

The following table lists event group names and hexadecimal values that you can use to specify that you want to get information about specific groups of MS-MPI APIs when you start tracing. If you use the **logman** command to start tracing, you can use the names in the **Group Name** column when you use the **-p** option. To include all MPI-related events when you use the **logman** command, omit the flags from the **-p** option. You can use the hexadecimal values with either the **logman** or **xperf** command. These event groups are defined in the mpitrace.man file that resides in the *%CCP\_HOME%\bin\* folder.

|  |  |  |
| --- | --- | --- |
| **Group Name** | **Description** | **Hexadecimal Equivalent** |
|  | All APIs | 0x0000000000007FFF |
| **mpi:p2p** | Point to point APIs | 0x0000000000000001 |
| **mpi:poll** | Point to point polling APIs such as MPI\_Iprobe and MPI\_Test*XXX* | 0x0000000000000002 |
| **mpi:coll** | Collective APIs | 0x0000000000000004 |
| **mpi:rma** | One-sided APIs | 0x0000000000000008 |
| **mpi:comm** | Communication APIs | 0x0000000000000010 |
| **mpi:eh** | Error handler APIs | 0x0000000000000020 |
| **mpi:grp** | Group APIs | 0x0000000000000040 |
| **mpi:attr** | Attribute APIs | 0x0000000000000080 |
| **mpi:dt** | Data type APIs | 0x0000000000000100 |
| **mpi:io** | Input/output APIs | 0x0000000000000200 |
| **mpi:topo** | Topology APIs | 0x0000000000000400 |
| **mpi:spwn** | Dynamic process APIs | 0x0000000000000800 |
| **mpi:init** | Initialization APIs | 0x0000000000001000 |
| **mpi:info** | Informational APIs | 0x0000000000002000 |
| **mpi:misc** | Miscellaneous APIs | 0x0000000000004000 |
|  | All interconnectivity communication | 0x00000000000F8000 |
| **mpi:sock** | Sockets interconnectivity communication | 0x0000000000008000 |
| **mpi:shm** | Shared memory interconnectivity communication | 0x0000000000010000 |
| **mpi:nd** | NetworkDirect interconnectivity communication | 0x0000000000020000 |
| **mpi:msg** | Mid-level message events that show the internal messages sent during collective operations | 0x0000000000040000 |
| **mpi:net\_rdata** | Events that receive data | 0x0000010000000000 |
| **mpi:net\_sdata** | Events that send data | 0x0000020000000000 |
| **mpi:api\_enter** | The start of an API routine | 0x0000200000000000 |
| **mpi:api\_leave** | The end of an API routine | 0x0000400000000000 |
| **mpi:api\_error** | An API error. This bit is set on the leave event for a function. | 0x0000800000000000 |
| **Microsoft-HPC-MPI/Api Windows HPC MPI Api Event Channel** | A channel that provides all events for all top-level APIs. | 0x4000000000000000 |
| **Microsoft-HPC-MPI/Communication Windows HPC MPI communication.Event Channel** | A channel that contains all transport event messages that occurred inside the API calls. | 0x8000000000000000 |

# Appendix 2: Related Information

* [Overview of ETW Tracing](http://msdn.microsoft.com/en-us/library/aa363668(VS.85).aspx)

http://msdn.microsoft.com/en-us/library/aa363668(VS.85).aspx

* [xperf documentation](http://msdn.microsoft.com/en-us/library/ff191081(VS.85).aspx)

http://msdn.microsoft.com/en-us/library/ff191081(VS.85).aspx

* [logman documentation](http://technet.microsoft.com/en-us/library/cc753820(WS.10).aspx)

http://technet.microsoft.com/en-us/library/cc753820(WS.10).aspx

* [MPICH2 Home Page](http://go.microsoft.com/fwlink/?LinkId=55115)

http://go.microsoft.com/fwlink/?LinkId=55115

* [MPI tutorial at the Lawrence Livermore National Lab](http://go.microsoft.com/fwlink/?LinkId=56096)

http://go.microsoft.com/fwlink/?LinkId=56096

* [Migrating Parallel Applications](http://go.microsoft.com/fwlink/?LinkId=55931)

http://go.microsoft.com/fwlink/?LinkId=55931

* [Debugging Parallel Applications Using Visual Studio 2005](http://go.microsoft.com/fwlink/?LinkId=55932)

http://go.microsoft.com/fwlink/?LinkId=55932