



A Comparison of Performance Analysis Tools on the NERSC SP

Jonathan Carter
NERSC User Services

Performance Tools on the IBM SP

- PE Benchmarker
 - IBM PSSP
 - Trace and visualize hardware counters values or MPI and user-defined events
- Paraver
 - European Center for Parallelism at Barcelona (CEPBA)
 - Trace and visualize program states, hardware counters values, and MPI and user-defined events
- Vampir
 - Pallas GmbH
 - Trace and visualize MPI and user-defined events
- PAPI, HPMLib, poe+, etc.
 - Tools based on pmapi hardware-counter interface
 - Summaries over entire execution

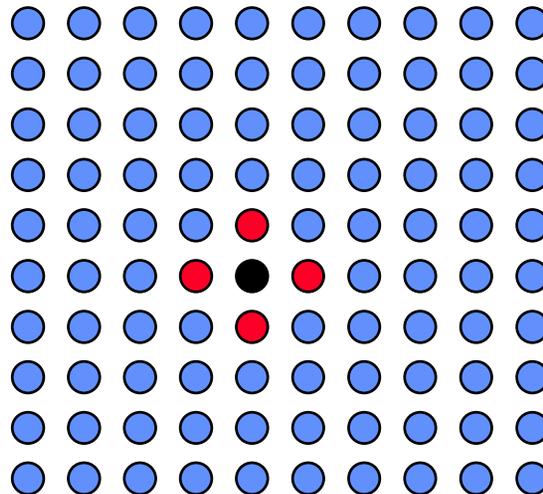
Example Program

- Solve Poisson equation on a 2D grid using Jacobi iteration

$$\nabla^2 u = f(x, y)$$

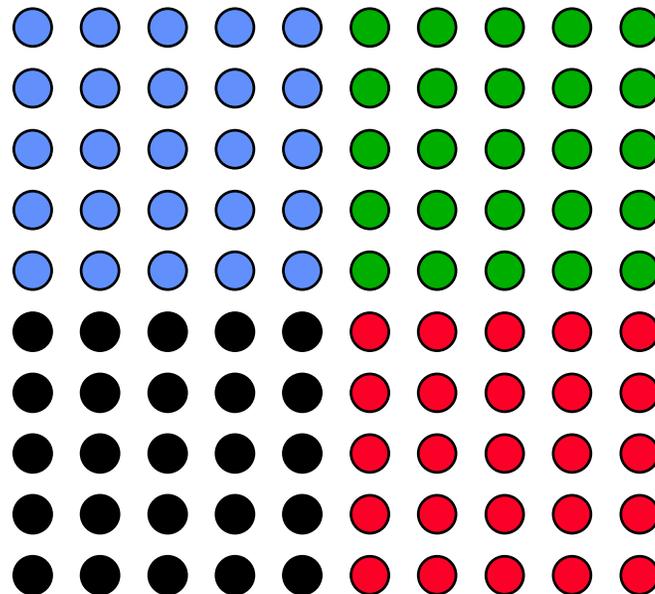
$$u(x, y) = g(x, y)$$

$$u_{i,j}^{k+1} = \frac{1}{4} (u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k + u_{i+1,j}^k - h^2 f_{i,j})$$



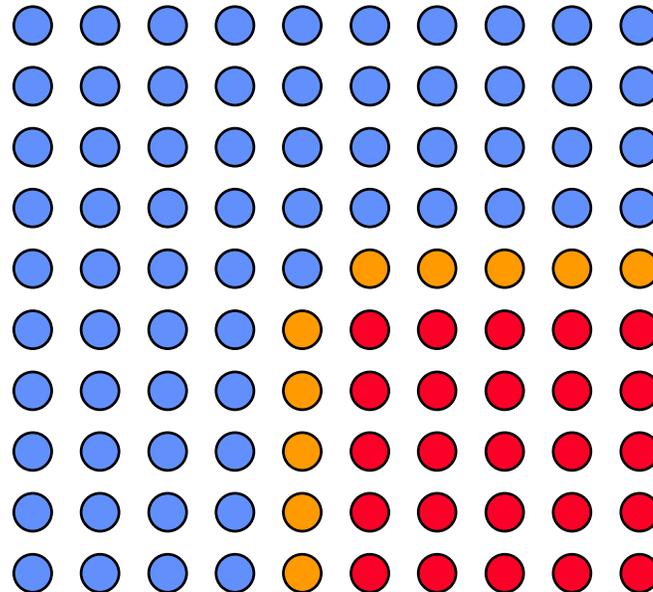
Example Program

- Decompose 2D grid into blocks, giving a block to each MPI process
- Assuming 4 MPI processes:



Example Program

- At the edges, information must be exchanged between MPI processes:



Example Program

- Flow chart of execution:

```
initialize
while ! converged
    exchange boundary data
    jacobi
    collect global sum of differences
    update
finalize
```

- Each MPI process is parallelized using OpenMP for the `jacobi` and `update` phases.

PE Benchmarker

- Performance Collection Tool (pct)
 - Application is run under the control of pct, uses DPCL to insert probes into executable before it runs.
 - Collects hardware/OS profiles at a thread level on a routine by routine basis
 - Collects MPI event statistics
- Performance Visualization Tool (pvt)
 - Graphical display of execution statistics
- MPI events
 - Summaries only (utestats), relies on ANL *jumpshot* to visualize MPI timelines

Performance Collection Tool

- Available with a graphical or command line interface
- May collect MPI event statistics or hardware/OS profiles (hardware counters, cpu time, etc.)
- Limitations:
 - Instrumentation takes place after program has been launched by POE. For a large number of source files and routines, this can take a considerable amount of time.
 - Requires 512 MB of shared memory segments

Performance Collection Tool

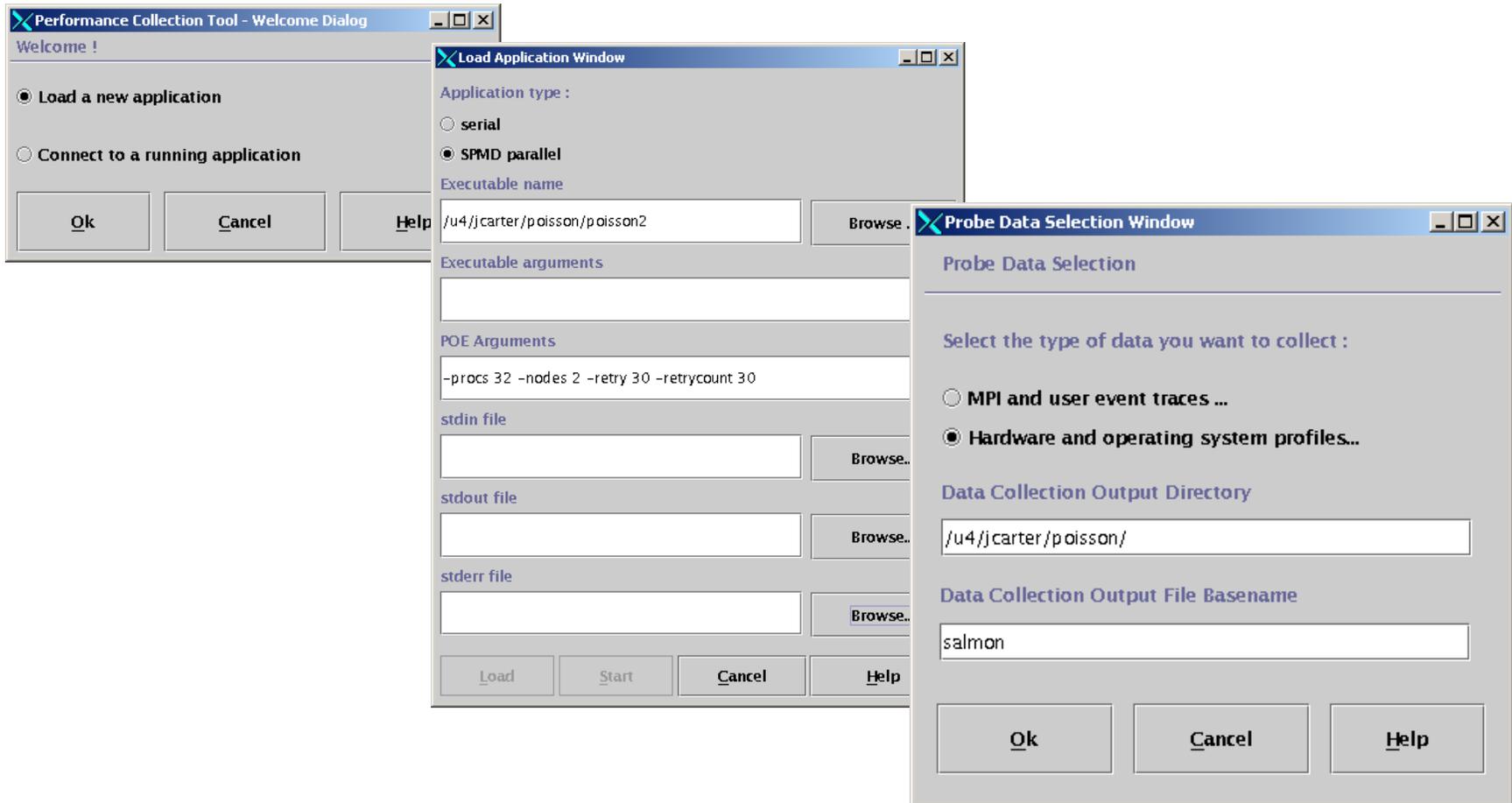
- For MPI event statistics
 - Select subset of MPI processes
 - Select subset of source files or routines
 - Select subset of MPI events
 - Add user-defined events
- For hardware/OS profiles
 - Select subset of processes
 - Select subset of timers, memory usage, hardware-counter group
 - Define hardware-counter groups (several already predefined)

Performance Collection Tool

- Must use "threaded" compilers for MPI tracing
- Must set environment variable `MP_UTE=YES` before linking MPI application if tracing of MPI events is required
- To run

```
module load java
pct
```

Performance Collection Tool

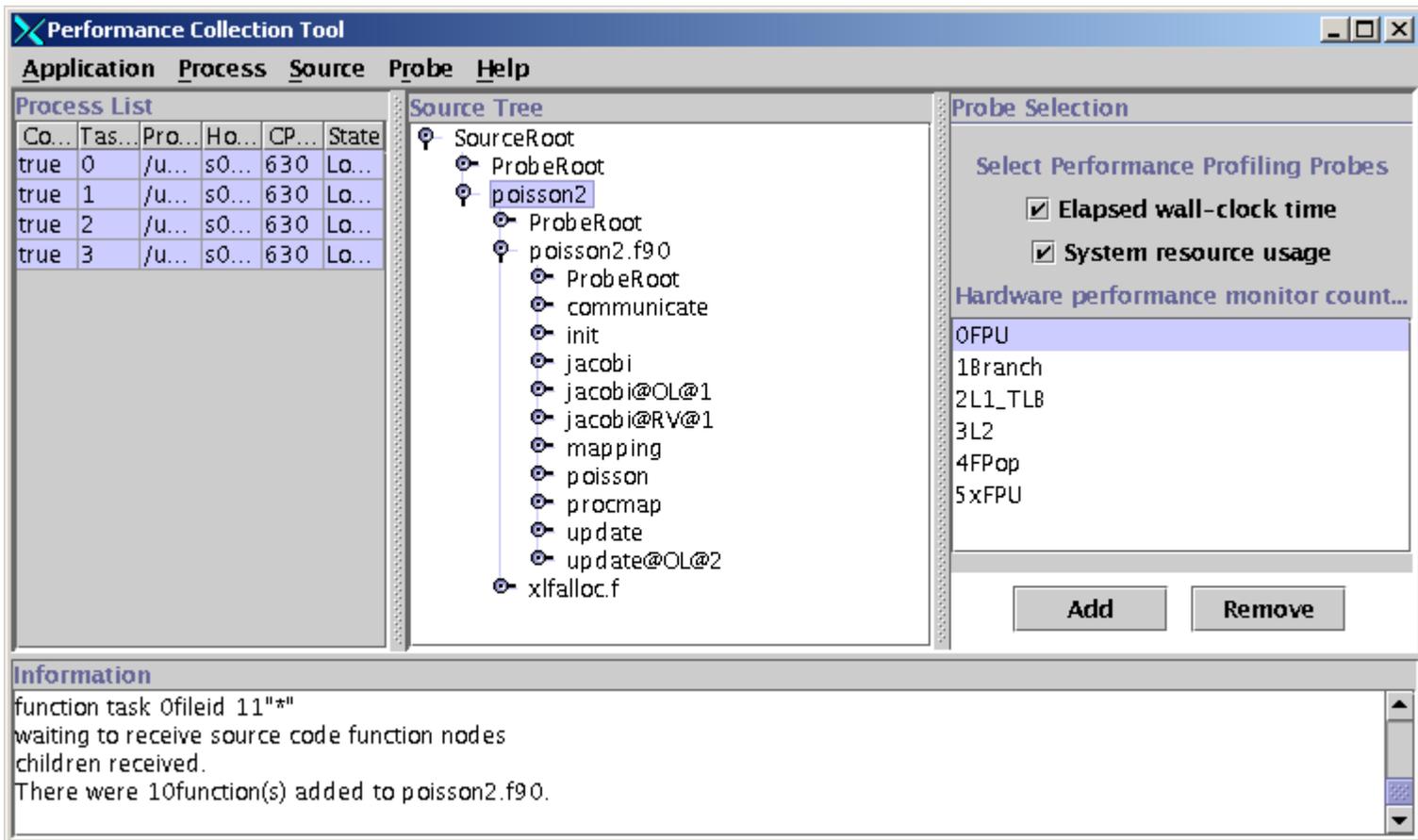


The image shows three overlapping dialog boxes from the Performance Collection Tool:

- Performance Collection Tool - Welcome Dialog:** Contains two radio buttons: "Load a new application" (selected) and "Connect to a running application". Buttons: Ok, Cancel, Help.
- Load Application Window:** Fields include:
 - Application type: serial, SPMD parallel
 - Executable name: /u4/jcarter/poisson/poisson2
 - Executable arguments: (empty)
 - POE Arguments: -procs 32 -nodes 2 -retry 30 -retrycount 30
 - stdin file, stdout file, stderr file: (empty)Buttons: Load, Start, Cancel, Help.
- Probe Data Selection Window:** Fields include:
 - Probe Data Selection: MPI and user event traces ..., Hardware and operating system profiles...
 - Data Collection Output Directory: /u4/jcarter/poisson/
 - Data Collection Output File Basename: salmonButtons: Ok, Cancel, Help.

Performance Collection Tool – Profiles

Select processes, routines, data to be collected:



The screenshot shows the Performance Collection Tool interface with the following components:

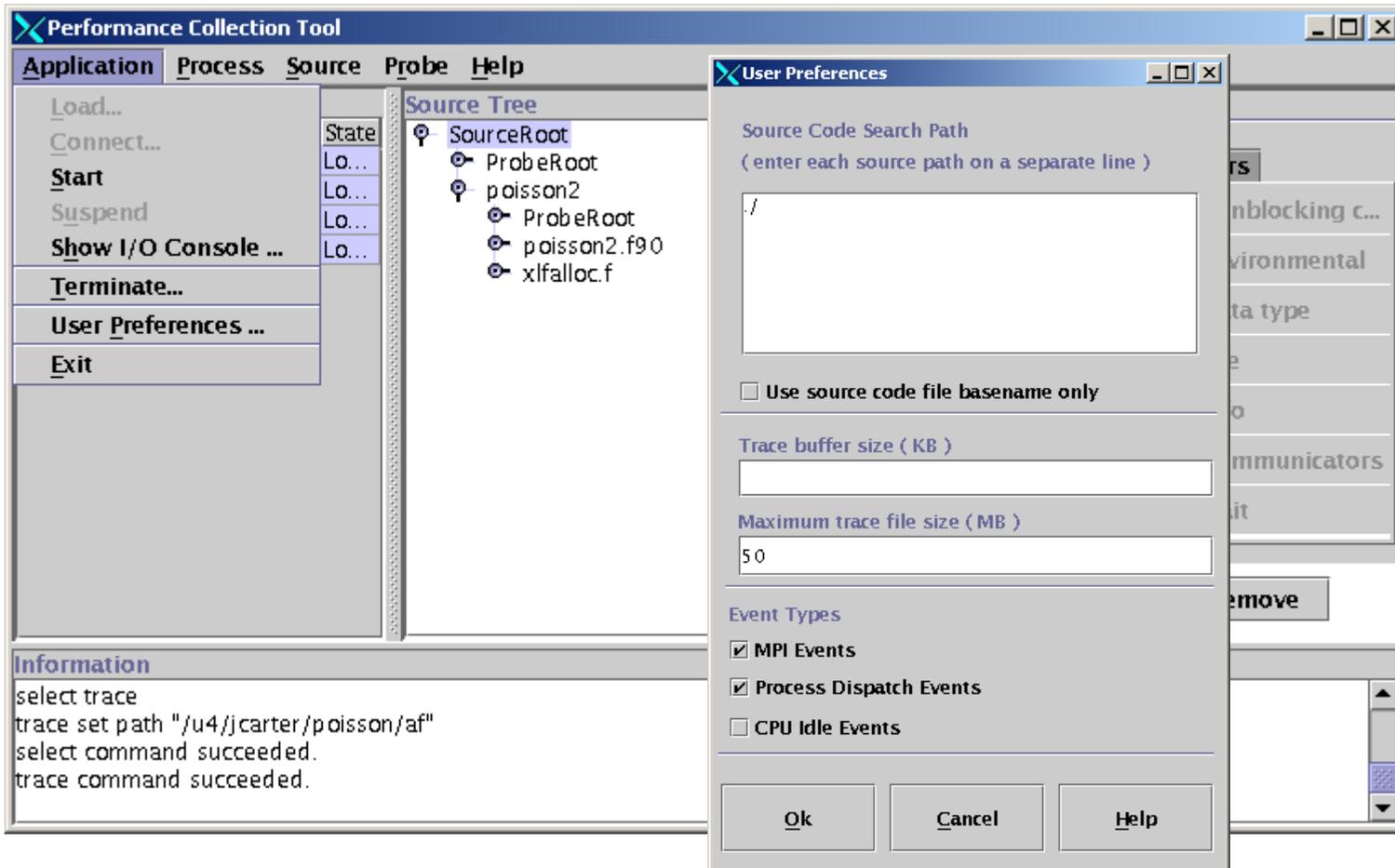
- Process List:** A table with columns for Co., Tas., Pro., Ho., CP., and State. It lists four processes with IDs 0, 1, 2, and 3.
- Source Tree:** A hierarchical tree view showing the structure of source code. The selected process is 'poisson2', which contains sub-routines like 'poisson2.f90', 'communicate', 'init', 'jacobi', and 'update'.
- Probe Selection:** A panel for selecting performance profiling probes. Two probes are checked: 'Elapsed wall-clock time' and 'System resource usage'. A list of hardware performance monitor counts is also visible, including OFPU, 1Branch, 2L1_TLB, 3L2, 4FPop, and 5xFPU.
- Information:** A text area at the bottom providing status updates, such as 'waiting to receive source code function nodes' and 'There were 10function(s) added to poisson2.f90'.

Performance Collection Tool – Profiles

- After the application completes:
 - Files named *basename.cdf.xx* created, one per process
 - Contain hardware profiles that can be viewed with the Performance Visualization Tool
 - Generally files are not too large

Performance Collection Tool – MPI Events

May need to increase Maximum trace file size, decide on Event Types to be monitored:



The screenshot shows the Performance Collection Tool interface with the User Preferences dialog box open. The main window has a menu bar with 'Application', 'Process', 'Source', 'Probe', and 'Help'. The 'Application' menu is open, showing options like 'Load...', 'Connect...', 'Start', 'Suspend', 'Show I/O Console ...', 'Terminate...', 'User Preferences ...', and 'Exit'. The 'Source Tree' pane shows a hierarchy: SourceRoot, ProbeRoot, poisson2, ProbeRoot, poisson2.f90, and xlfalloc.f. The 'User Preferences' dialog has the following settings:

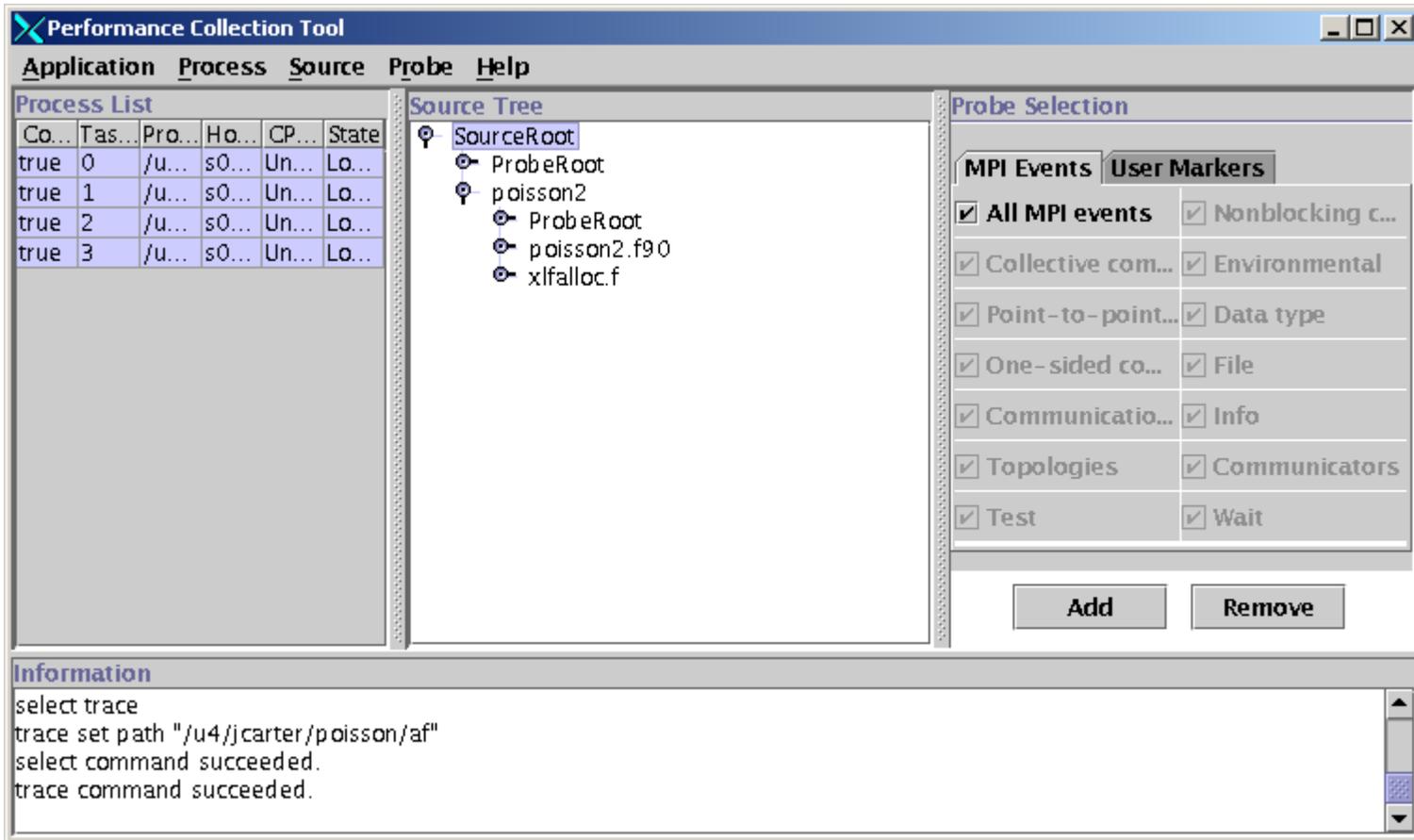
- Source Code Search Path: (enter each source path on a separate line) with a text area containing '/'
- Use source code file basename only
- Trace buffer size (KB): [empty text box]
- Maximum trace file size (MB): 50
- Event Types:
 - MPI Events
 - Process Dispatch Events
 - CPU Idle Events

The 'Information' pane at the bottom left shows the following text:

```
select trace
trace set path "/u4/jcarter/poisson/af"
select command succeeded.
trace command succeeded.
```

Performance Collection Tool – MPI Events

Selecting MPI Events:



The screenshot shows the Performance Collection Tool interface with the following components:

- Process List:** A table with columns: Co..., Tas..., Pro..., Ho..., CP..., State. It contains four rows of process information.
- Source Tree:** A tree view showing the source code structure, including SourceRoot, ProbeRoot, poisson2, and xlfalloc.f.
- Probe Selection:** A panel with two tabs: MPI Events and User Markers. The MPI Events tab is active, showing a list of event categories with checkboxes. The User Markers tab is also visible.
- Information:** A text area at the bottom showing the command history for selecting the trace and setting the path.

Co...	Tas...	Pro...	Ho...	CP...	State
true	0	/u...	s0...	Un...	Lo...
true	1	/u...	s0...	Un...	Lo...
true	2	/u...	s0...	Un...	Lo...
true	3	/u...	s0...	Un...	Lo...

Source Tree

- SourceRoot
 - ProbeRoot
 - poisson2
 - ProbeRoot
 - poisson2.f90
 - xlfalloc.f

Probe Selection

MPI Events **User Markers**

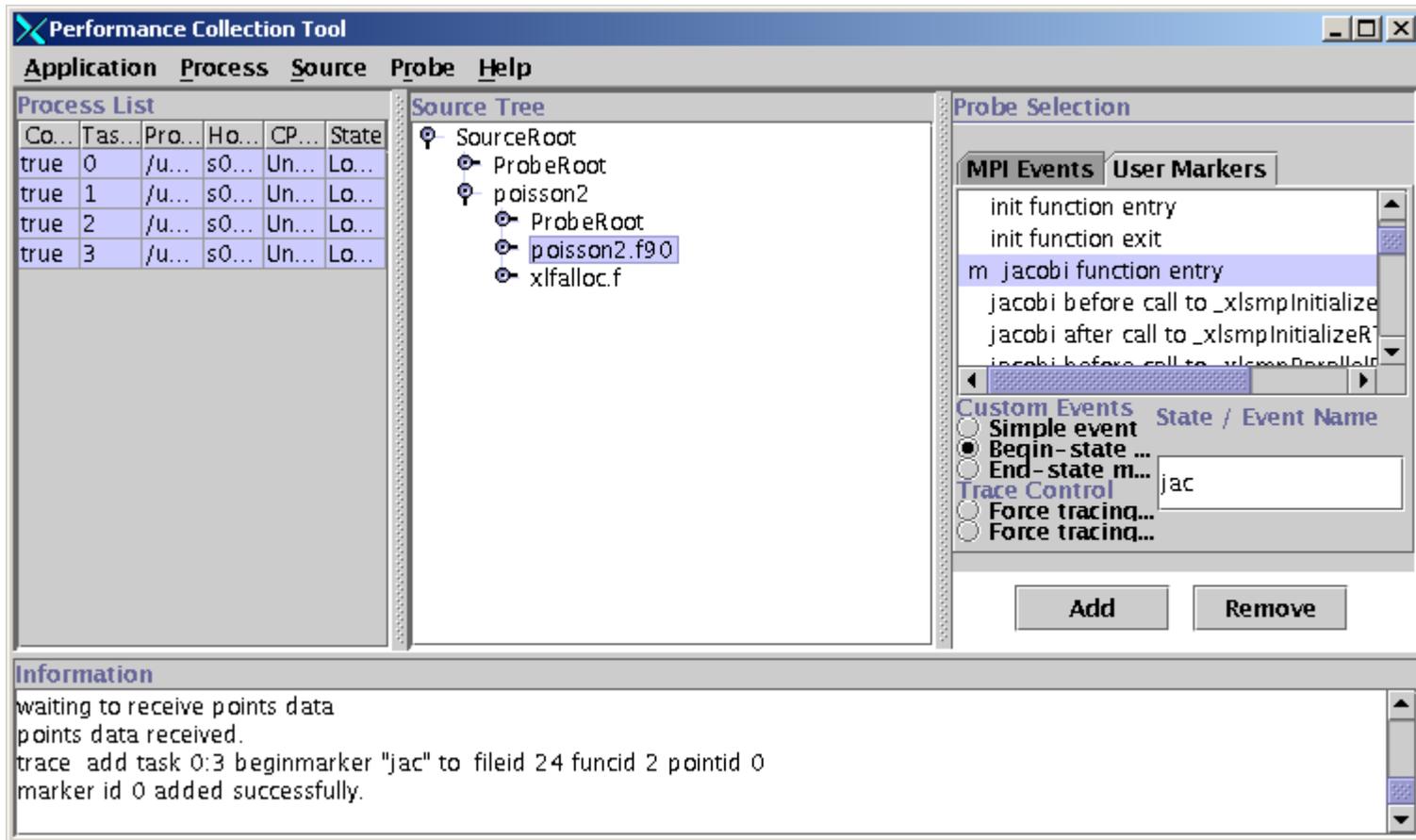
- All MPI events
- Collective com...
- Point-to-point...
- One-sided co...
- Communicatio...
- Topologies
- Test
- Nonblocking c...
- Environmental
- Data type
- File
- Info
- Communicators
- Wait

Information

```
select trace
trace set path "/u4/jcarter/poisson/af"
select command succeeded.
trace command succeeded.
```

Performance Collection Tool – MPI Events

Selecting User Markers:



The screenshot shows the Performance Collection Tool interface. The 'Probe Selection' panel is active, with the 'User Markers' tab selected. The list of markers includes 'm_jacobi function entry', which is currently selected. Below the list, the 'Custom Events' section has 'Begin-state ...' selected, and the 'State / Event Name' field contains the text 'jac'. The 'Add' and 'Remove' buttons are visible at the bottom of the panel.

Process List

Co...	Tas...	Pro...	Ho...	CP...	State
true	0	/u...	s0...	Un...	Lo...
true	1	/u...	s0...	Un...	Lo...
true	2	/u...	s0...	Un...	Lo...
true	3	/u...	s0...	Un...	Lo...

Source Tree

- SourceRoot
 - ProbeRoot
 - poisson2
 - ProbeRoot
 - poisson2.f90
 - xlalloc.f

Probe Selection

MPI Events | **User Markers**

- init function entry
- init function exit
- m_jacobi function entry
- jacobi before call to_xlsmplInitialize
- jacobi after call to_xlsmplInitializeR
- jacobi before call to_xlsmplRevelSelf

Custom Events | **State / Event Name**

- Simple event
- Begin-state ...
- End-state m...

Trace Control

- Force tracing...
- Force tracingq...

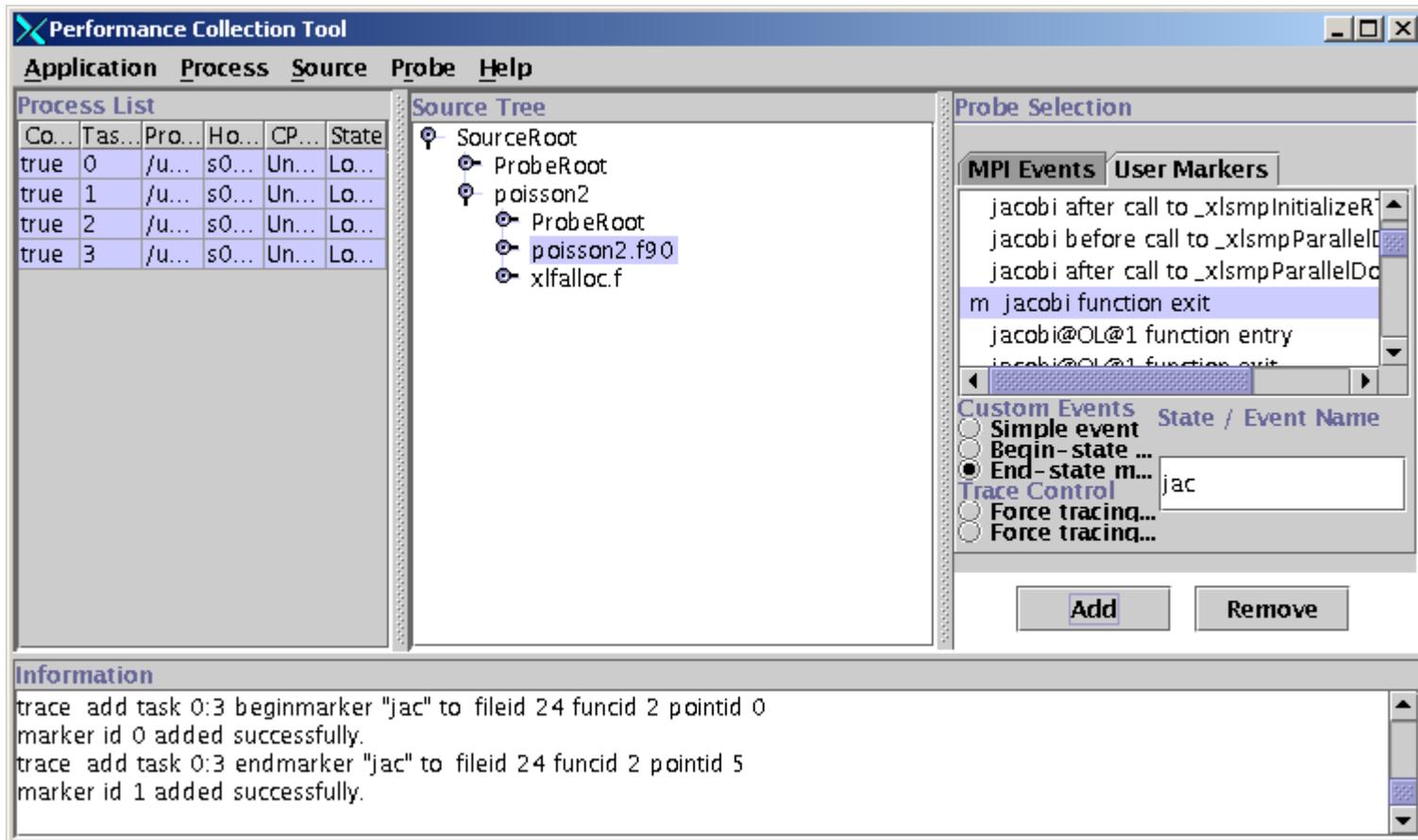
jac

Add Remove

Information

```
waiting to receive points data
points data received.
trace add task 0:3 beginmarker "jac" to fileid 24 funcid 2 pointid 0
marker id 0 added successfully.
```

Performance Collection Tool – MPI Events



The screenshot shows the Performance Collection Tool interface with the following components:

- Process List:** A table with columns for Co..., Tas..., Pro..., Ho..., CP..., and State. It lists four processes (0, 1, 2, 3) all in a 'Lo...' state.
- Source Tree:** A tree view showing the source code structure, including SourceRoot, ProbeRoot, poisson2, and xlfalloc.f.
- Probe Selection:** A panel with tabs for MPI Events and User Markers. The MPI Events list includes 'jacobi after call to _xlsmpParallelDc' and 'm jacobi function exit'. The User Markers section has a text input field containing 'jac'.
- Information:** A log area at the bottom showing messages like 'trace add task 0:3 beginmarker "jac" to fileid 24 funcid 2 pointid 0'.

Performance Collection Tool – MPI Events

- After the application completes:
 - Files named *basename.xx* created, one per node (xx is rank of one random task per node)
 - Files are generally large (AIX trace files)
 - Depends on what is traced: mpi, process, idle
 - Need to preprocess files with the *uteconvert* command to produce UTE files
 - Can collect statistics directly from UTE files using *utestats* command, and use the *slogmerge* in order to view with *jumpshot*

Command Line Interface

- Invoke with `pct -c`
- Provide a script file with `pct -c -s scriptfile`

```
load poe exec /usr/common/homes/j/jcarter/poisson/poisson2 poeargs "-procs 4
-nodes 1 -retry 30 -retrycount 30"
select trace
trace set path "/usr/common/homes/j/jcarter/poisson/tmp"
trace set logsize 50
trace set event mpi
trace add mpiname all to file "*"
trace add beginmarker "jacobi" to file "poisson2.f90" funcid 2 pointid 0
trace add endmarker "jacobi" to file "poisson2.f90" funcid 2 pointid 5
trace add beginmarker "update" to file "poisson2.f90" funcid 8 pointid 0
trace add endmarker "update" to file "poisson2.f90" funcid 8 pointid 5
start
wait
```

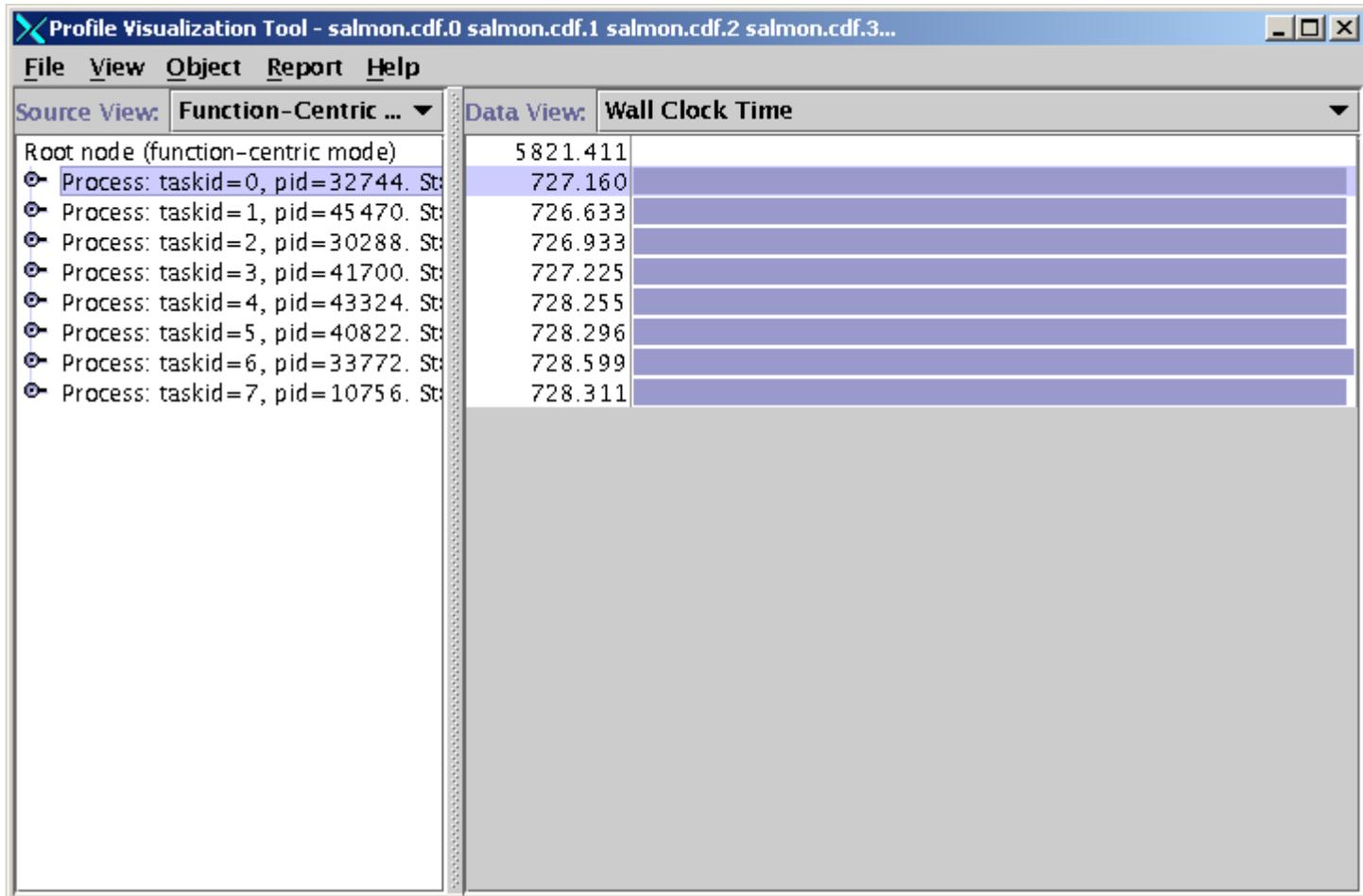
Performance Visualization Tool

- View hardware/OS profiles
- To run

```
module load java  
pvt basename.cdf.*
```

Performance Visualization Tool

Select data to be displayed from "Data View" menu:

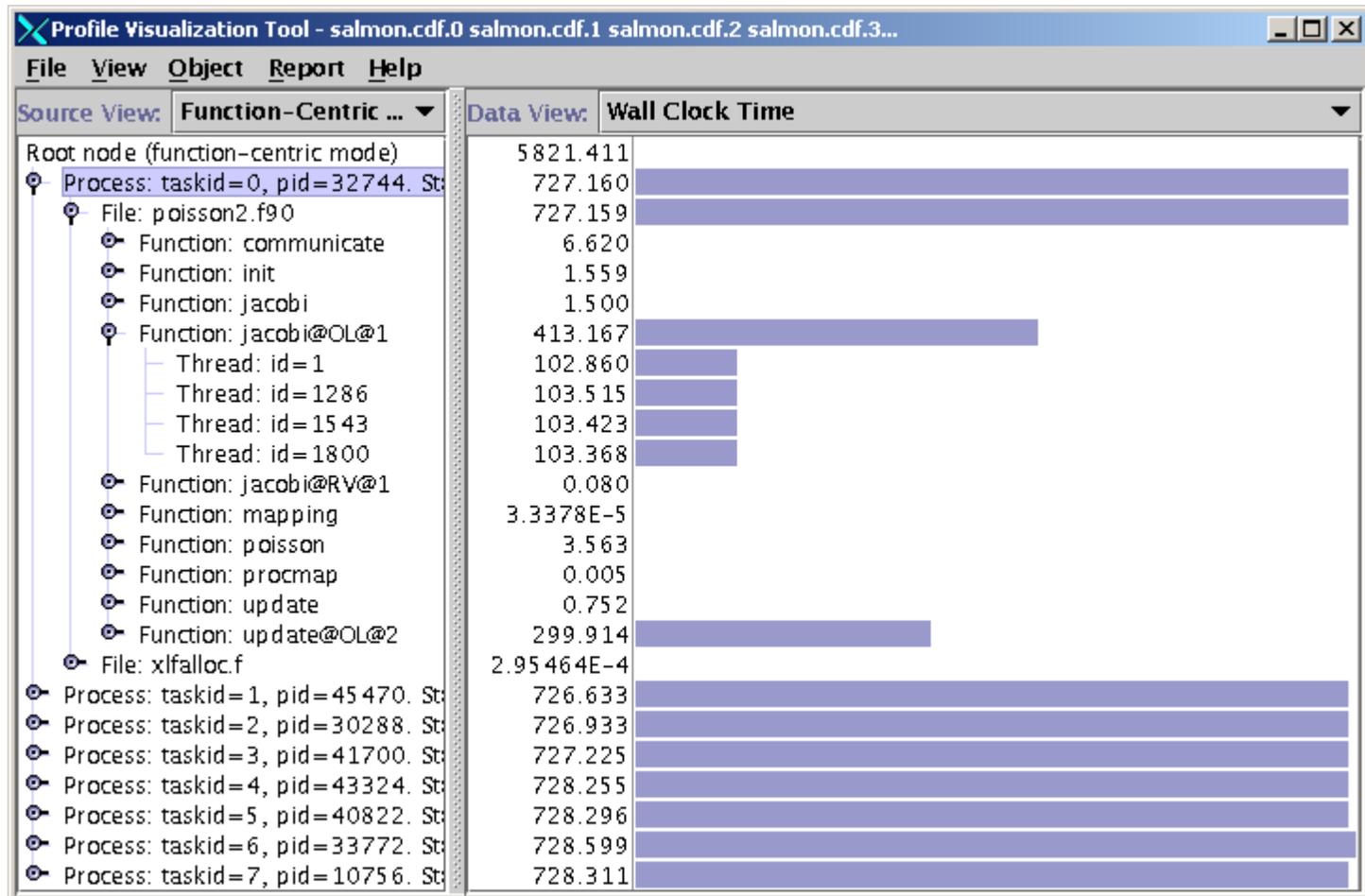


The screenshot shows the Profile Visualization Tool interface. The window title is "Profile Visualization Tool - salmon.cdf.0 salmon.cdf.1 salmon.cdf.2 salmon.cdf.3...". The menu bar includes "File", "View", "Object", "Report", and "Help". The "Data View" menu is open, showing "Wall Clock Time" selected. The main display area contains a table with the following data:

Source View	Function-Centric ...	Data View	Wall Clock Time
Root node (function-centric mode)			5821.411
Process: taskid=0, pid=32744. St			727.160
Process: taskid=1, pid=45470. St			726.633
Process: taskid=2, pid=30288. St			726.933
Process: taskid=3, pid=41700. St			727.225
Process: taskid=4, pid=43324. St			728.255
Process: taskid=5, pid=40822. St			728.296
Process: taskid=6, pid=33772. St			728.599
Process: taskid=7, pid=10756. St			728.311

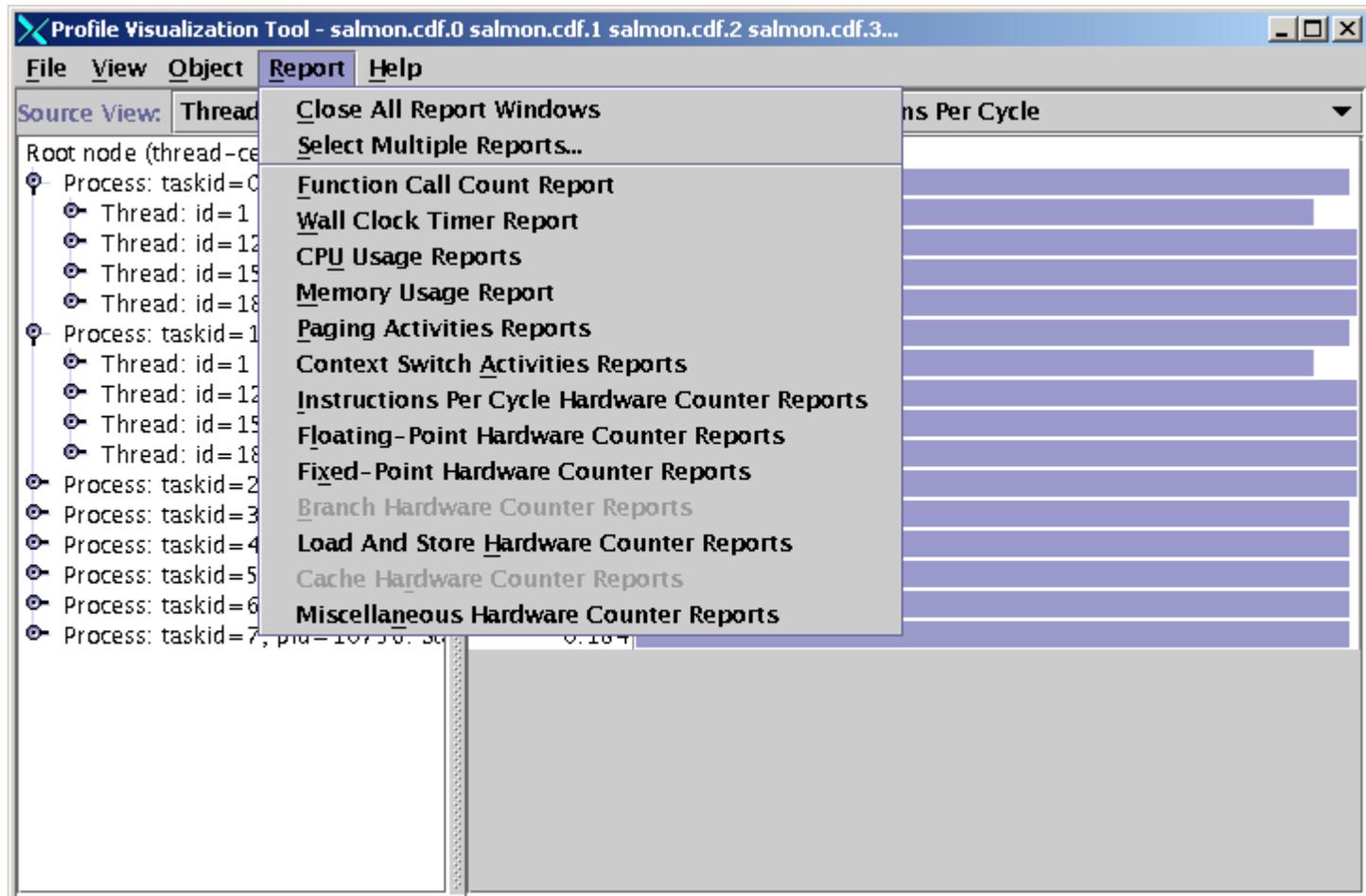
Performance Visualization Tool

Expand processes to show functions and threads:



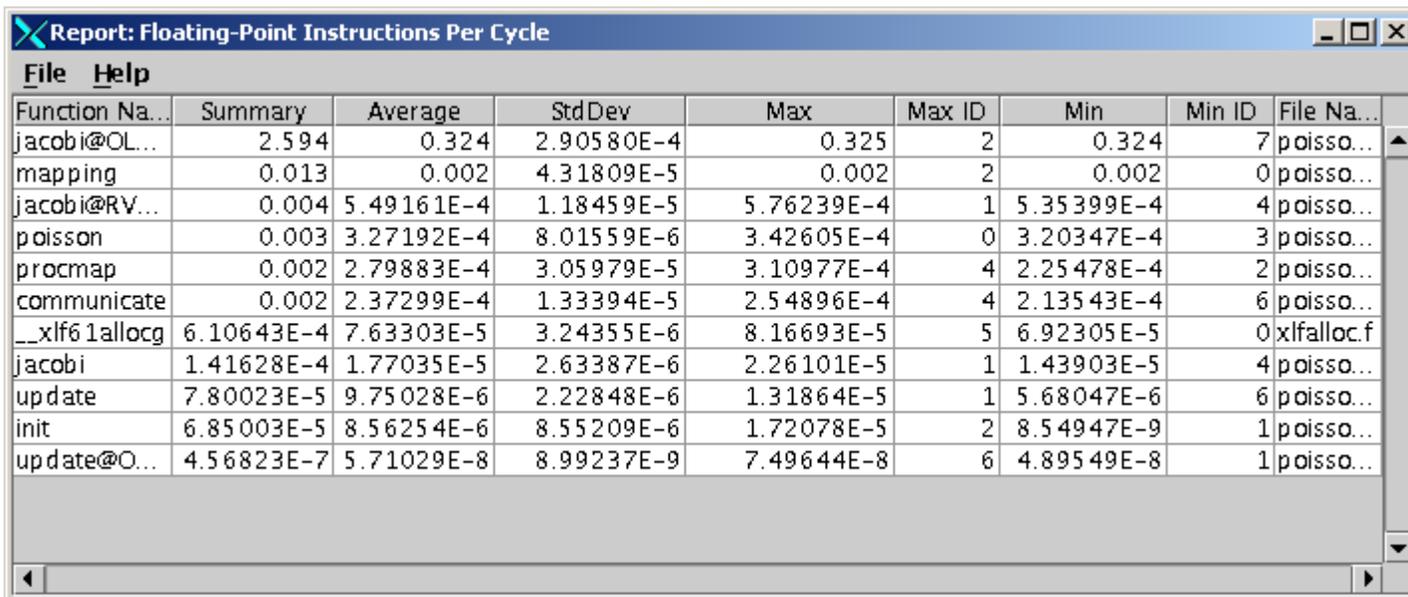
Performance Visualization Tool

Produce various text reports:



Performance Visualization Tool

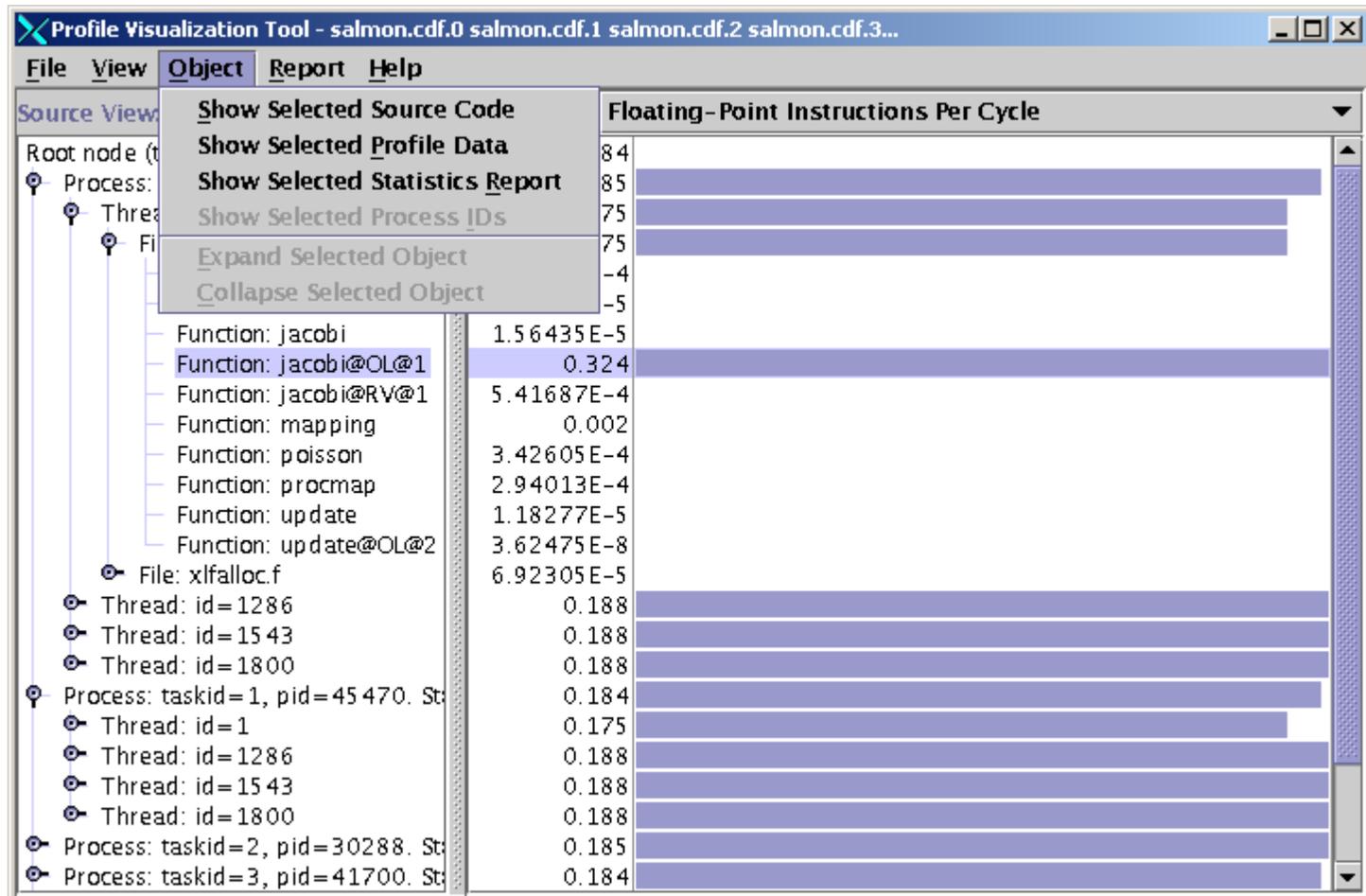
Text reports can be saved as tab-separated text files.



Function Na...	Summary	Average	StdDev	Max	Max ID	Min	Min ID	File Na...
jacobi@OL...	2.594	0.324	2.90580E-4	0.325	2	0.324	7	poisso...
mapping	0.013	0.002	4.31809E-5	0.002	2	0.002	0	poisso...
jacobi@RV...	0.004	5.49161E-4	1.18459E-5	5.76239E-4	1	5.35399E-4	4	poisso...
poisson	0.003	3.27192E-4	8.01559E-6	3.42605E-4	0	3.20347E-4	3	poisso...
procmmap	0.002	2.79883E-4	3.05979E-5	3.10977E-4	4	2.25478E-4	2	poisso...
communicate	0.002	2.37299E-4	1.33394E-5	2.54896E-4	4	2.13543E-4	6	poisso...
__xlf61allocg	6.10643E-4	7.63303E-5	3.24355E-6	8.16693E-5	5	6.92305E-5	0	xlfalloc.f
jacobi	1.41628E-4	1.77035E-5	2.63387E-6	2.26101E-5	1	1.43903E-5	4	poisso...
update	7.80023E-5	9.75028E-6	2.22848E-6	1.31864E-5	1	5.68047E-6	6	poisso...
init	6.85003E-5	8.56254E-6	8.55209E-6	1.72078E-5	2	8.54947E-9	1	poisso...
update@O...	4.56823E-7	5.71029E-8	8.99237E-9	7.49644E-8	6	4.89549E-8	1	poisso...

Performance Visualization Tool

Show detailed statistics on one process, function or thread:



Performance Visualization Tool

Function Statistics Report: jacobi@OL@1								
...	this	Summary	Average	StdDev	Max	Max ID	Min	Min ID
Function Call ...	500	16000	2000	0	2000	0	2000	0
Wall Clock Time	102.860	3306.504	413.313	0.578	414.121	4	412.340	2
User CPU Usage	101.660	3263.150	407.894	0.272	408.460	7	407.590	5
System CPU U...	0.930	31.860	3.983	0.247	4.410	6	3.640	2
Maximum Res...	300228	2401464	300183	17.176	300228	0	300172	5
Page Fault Wit...	6114	195592	24449	4.330	24456	5	24442	7
Page Fault Wit...	0	0	0	0	0	0	0	0
Voluntary Con...	0	0	0	0	0	0	0	0
Involuntary Co...	33	26337	3292.125	647.622	4651	4	2488	3
Floating-Point...	0.324	2.594	0.324	2.90580E-4	0.325	2	0.324	7
Fixed-Point In...	0.004	0.035	0.004	2.12258E-4	0.005	5	0.004	2
Branch Instruc...	N/A							
Load Instructi...	0.163	1.306	0.163	1.21356E-4	0.163	2	0.163	7
Store Instructi...	0.041	0.329	0.041	3.06346E-5	0.041	4	0.041	3
Number Of Lo...	6.28826E9	2.01311E11	2.51639E10	8170277.127	2.51748E10	5	2.51546E10	0
FPU1 (Floating...	6.00129E9	1.92042E11	2.40052E10	505732.519	2.40058E10	5	2.40045E10	0
Number Of St...	1.58338E9	5.07191E10	6.33988E9	4307560.447	6.34556E9	5	6.33485E9	0
Processor Clo...	3.85545E10	1.23337E12	1.54171E11	1.38131E8	1.54408E11	7	1.53941E11	2
FPU0 (Floating...	6.49821E9	2.07942E11	2.59928E10	497611.849	2.59935E10	0	2.59922E10	5
FXU2 (Fixed-...	1.88941E7	6.39244E8	7.99056E7	3856520.521	8.49423E7	5	7.54959E7	2
FXU0 (Fixed-...	9.57332E7	2.95926E9	3.69907E8	6.52351E7	4.32752E8	5	2.45477E8	2
FXU1 (Fixed-...	4.59100E7	1.82913E9	2.28641E8	5.93726E7	3.42402E8	4	1.83807E8	0

Visualizing MPI Events

- Convert AIX tracefiles to UTE tracefiles

```
uteconvert -n n basename.
```

- Merge UTE tracefiles

```
utemerge -n n basename.ute.
```

- Generate statistics file

```
utestats -o basename.stats basename.ute.ute
```

- Generate slog file for jumpshot tool

```
slogmerge -n n basename.ute.
```

- Run jumpshot (limit is 64 MPI processes)

```
module load java mpe  
jumpshot basename.slog
```

jumpshot

Entire profile is divided into frames, select frame, select MPI-Process or Thread, then click "Display".



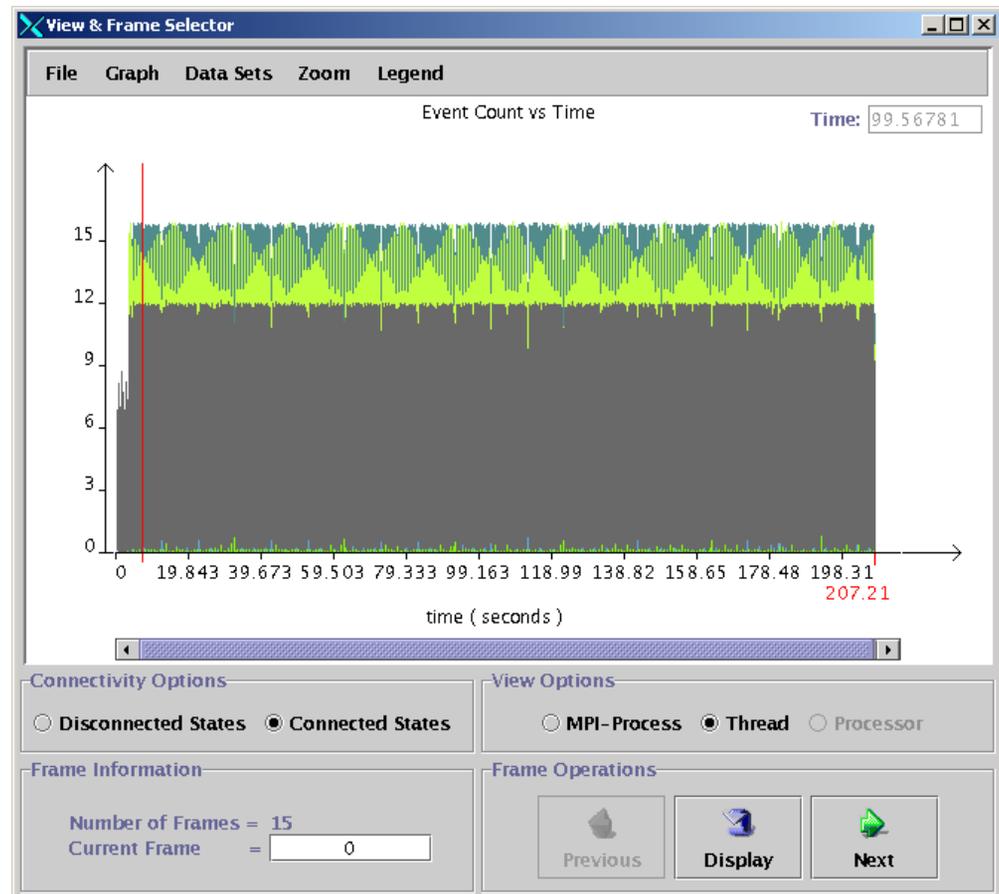
Legend

- MPI_Sendrecv
- MPI_Allreduce
- MPI_Comm_rank
- MPI_Comm_size
- MPI_Cart_create
- MPI_Cart_get
- MPI_Cart_shift
- MPI_Type_commit
- MPI_Type_contig...
- MPI_Type_vector
- Running**
- jacobi
- update
- Backward Arrow
- Forward Arrow

Select/Deselect

All None

Change Color



View & Frame Selector

File Graph Data Sets Zoom Legend

Event Count vs Time Time: 99.56781

15
12
9
6
3
0

0 19.843 39.673 59.503 79.333 99.163 118.99 138.82 158.65 178.48 198.31

time (seconds) 207.21

Connectivity Options
 Disconnected States Connected States

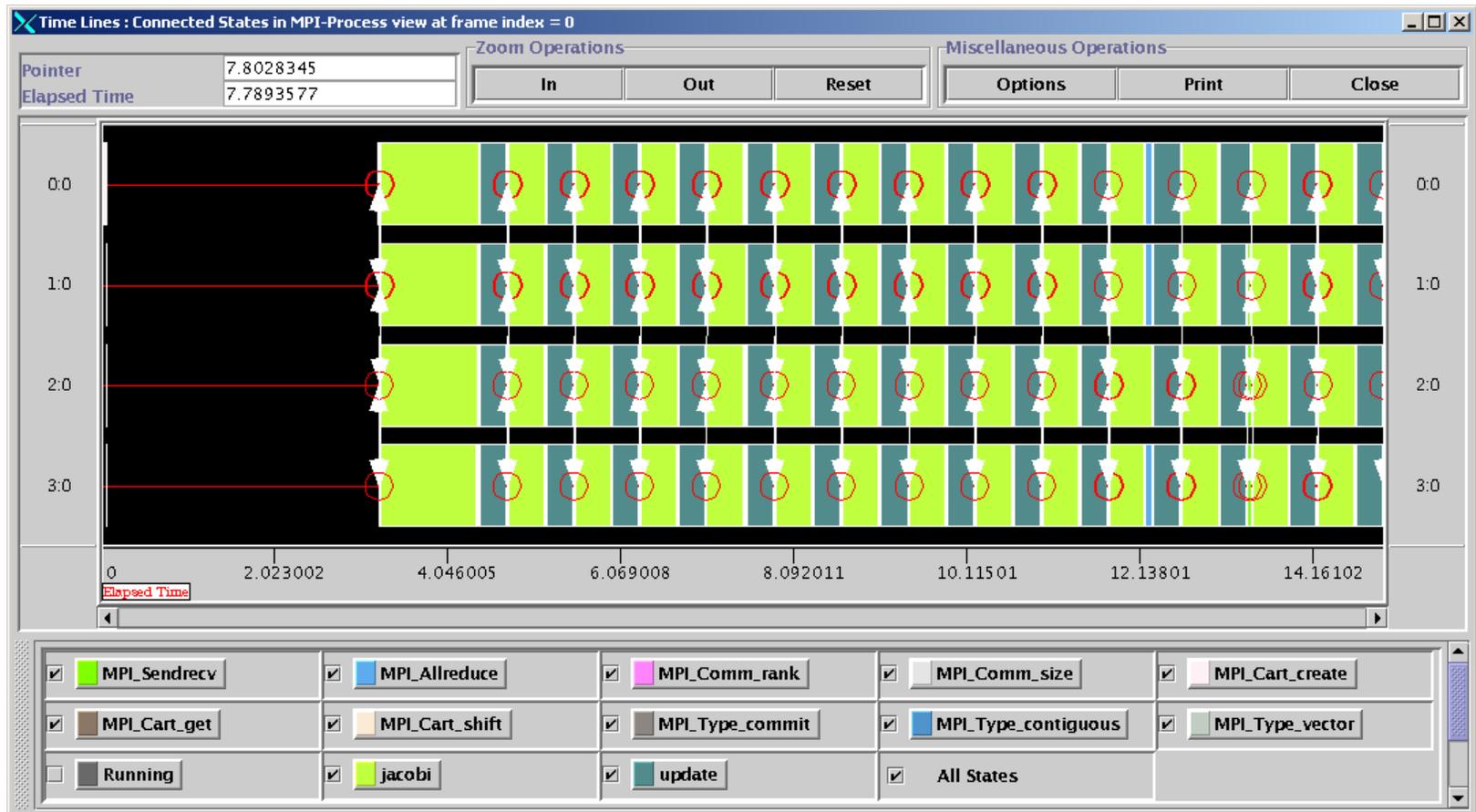
View Options
 MPI-Process Thread Processor

Frame Information
Number of Frames = 15
Current Frame = 0

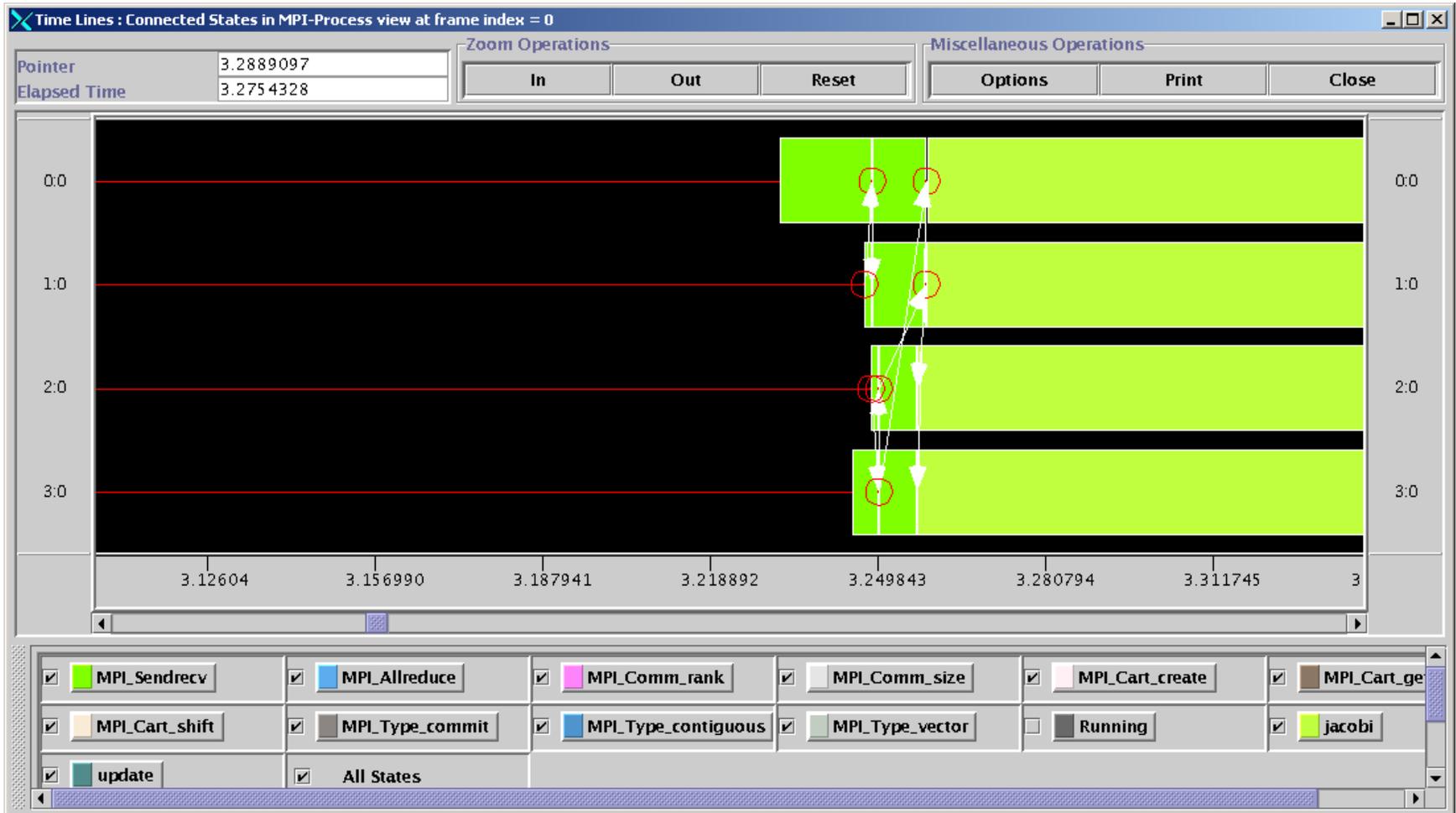
Frame Operations
Previous Display Next

jumpshot

Basic MPI process view, use zoom buttons to see detail

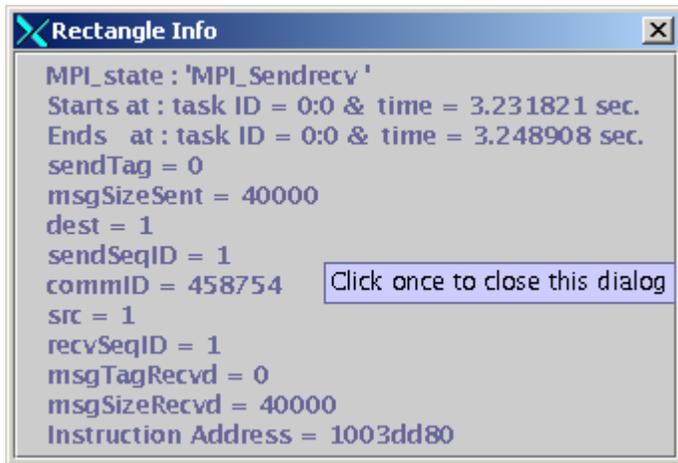


jumpshot



jumpshot

Clicking on messages (red circles) or blocks produces a detailed text description:



Rectangle Info

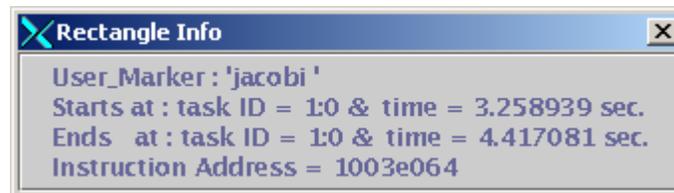
MPI_state : 'MPI_Sendrecv '
Starts at : task ID = 0:0 & time = 3.231821 sec.
Ends at : task ID = 0:0 & time = 3.248908 sec.
sendTag = 0
msgSizeSent = 40000
dest = 1
sendSeqID = 1
commID = 458754
src = 1
recvSeqID = 1
msgTagRecv = 0
msgSizeRecv = 40000
Instruction Address = 1003dd80

Click once to close this dialog



Arrow Info

Message : 'Backward Arrow '
Starts at : task ID = 0:0 & time = 3.248908 sec.
Ends at : task ID = 1:0 & time = 3.248866 sec.

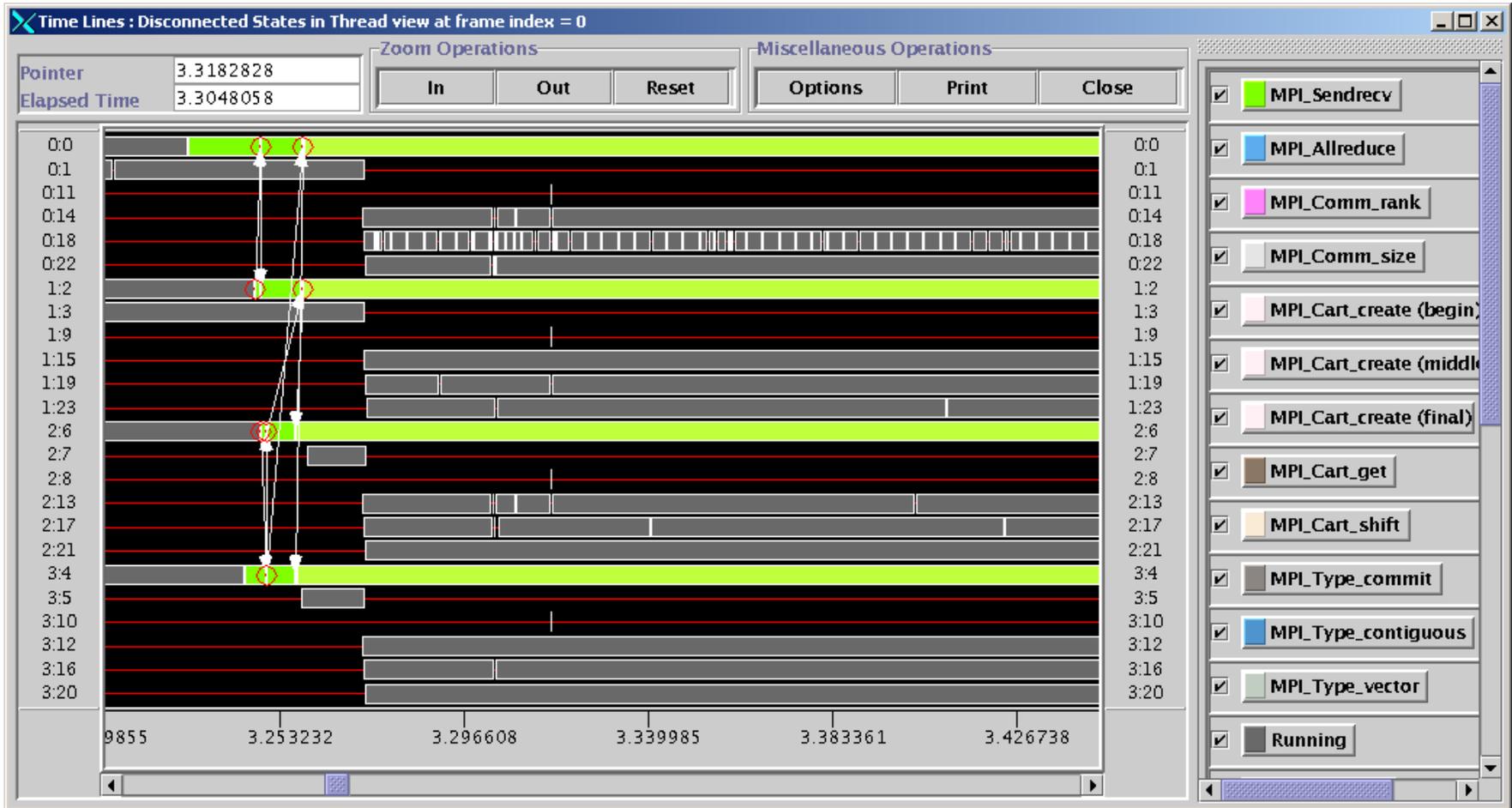


Rectangle Info

User_Marker : 'jacobi '
Starts at : task ID = 1:0 & time = 3.258939 sec.
Ends at : task ID = 1:0 & time = 4.417081 sec.
Instruction Address = 1003e064

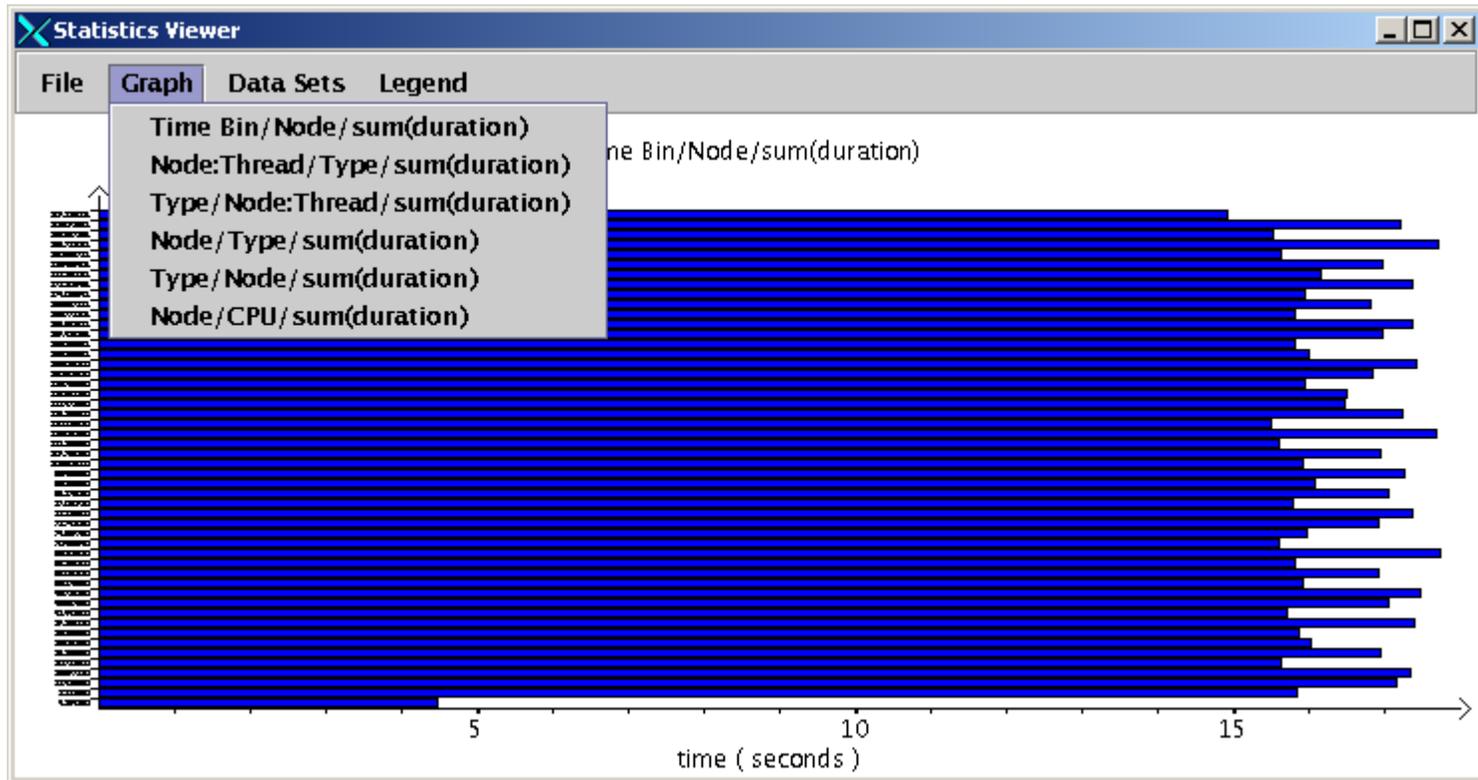
jumpshot

Thread level view:

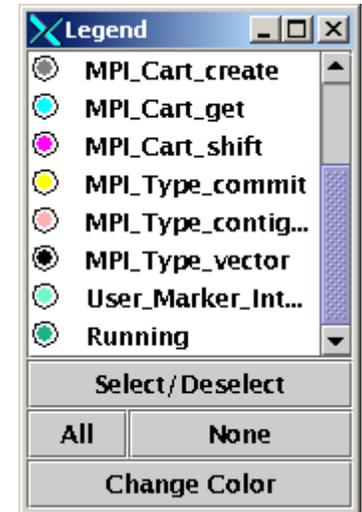
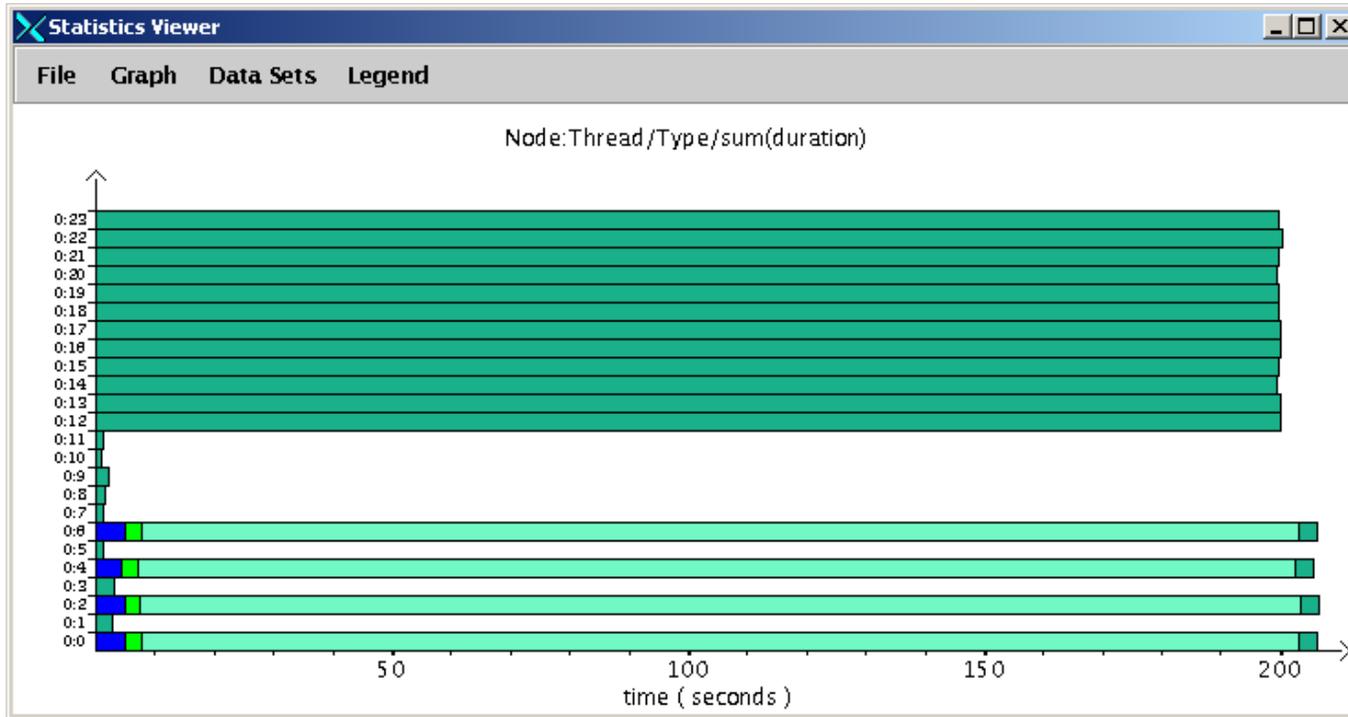


jumpshot

To visualize statistics file. Go back to "View and Frame Selector Window", use "File->View Statistics" menu to open the "Statistics Viewer" window, and from there open previously generated "stats" file.



jumpshot



Legend

- MPI_Cart_create
- MPI_Cart_get
- MPI_Cart_shift
- MPI_Type_commit
- MPI_Type_contig...
- MPI_Type_vector
- User_Marker_Int...
- Running

Select/Deselect

All None

Change Color

Further Reading

- IBM PE for AIX, Operation and Use, Volume 2
 - Chapter 3 and Appendix A
 - http://hpcf.nersc.gov/vendor_docs/ibm/pe/am103mst.html
- Tour of Jumpshot
 - `/usr/common/usg/mpe/1.2.2/share/jumpshot-3/doc/TourStepByStep.pdf`

CEPBA Tools

- Run application under the control of *ompttrace*
 - Uses DPCL technology to insert probes into executable
- Merge trace files with *omp2prv*
- Visualize results with *paraver*

omptrace

- `omptrace [-v] [-locks] [-nosw] [-r] [functions:[all|funcfile]]`
`[-calls:callfile] [-stdin:stdin] [-stdout:stdout]`
`[-stderr:stderr] [-counters] FullPathPOE`
`FullPathApplName [ApplPOEParams]`
- Options:
 - `-v` Verbose mode
 - `-locks` Trace lock calls
 - `-nosw` Not use the switch clock.
 - `-r` Must be set if application uses threaded MPI library
 - `-function` Extra functions to be traced (user code)
 - `-calls` Extra calls to be traced (system code)

omptirace

- -counters:[*option*,...] Trace counters automatically, options:
 - **user**/nouser
 - **parallel**/noparallel
 - **calls**/nocalls
 - mpi/**nompi**
- omptirace -list -functions:*funcfile* -calls:*callfile*
FullPathPOE FullPathApplName [ApplPOEParams]
 - -list List all traceable application functions/calls.
- omptirace -hwc [*counter*]
 - -hwc [*counter*] List the event counters available for each counter.
If a counter is selected, event are listed only for the selected ones.

Related environment variables

- `MPTRACE_ADDFUNCTIONS`: Filename containing additional functions to trace.
- `MPTRACE_DIR`: Output temporary/tracefile directory.
- `MPTRACE_BUFFER_SIZE`: Number of events allowed in thread event buffers.
- `MPTRACE_FILE_SIZE`: Maximum trace file size in Mbytes
- `MPTRACE_COUNTERS`: Hardware event traced for each hardware counter. Can use `$MPTRACE_FPHWC` to use same set as *hpmcount* command.
- `MPTRACE_LABELS`: User manually event traced labels.

omptirace

- ompitrace is a powerful tool to capture the states and events of a program:
 - Each thread passes through many states as it executes. The two basic states are *running* and *idle*. Also recognized are states that are associated with OpenMP (thread) or MPI actions, e.g. scheduling a thread, waiting for a message, collective operation.
 - Data can be recorded at certain execution events. Typical events are the entry and exit of a threaded-routine, the entry and exit of an MPI routine. You can add the entry and exit of any user-written routines. When an event occurs, data related to event are saved along with a time stamp.

ompttrace

```
$ module load cepba
$ export MPTRACE_COUNTERS=${MPTRACE_FPHWC}
$ ompitrace -r -nosw -counters:parallel poe poisson2 -procs 4 -nodes 1
OMPItrace tool (Version 1.1)
```

Tracing application: poisson2

Tracing Parameters:

- * Local clock
- * Tracing counters: user,parallel,calls,nompi

Parameters:

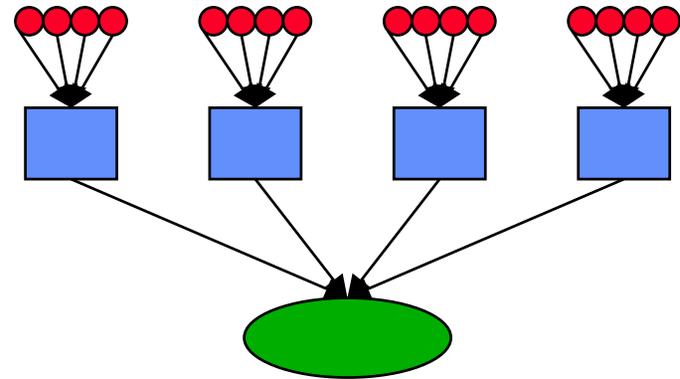
- 0: /usr/bin/poe
- 1: /usr/common/homes/j/jcarter/poisson/poisson2
- 2: -procs
- 3: 4
- 4: -nodes
- 5: 1

ompttrace

```
-> Creating the target application <-
subfilter: default repo mpccc will be charged
llsubmit: Processed command file through Submit Filter:
  "/usr/common/nsg/etc/subfilter".
-> Expanding source files <-
    -> ( 0 ) Expanding: poisson2.f90
    -> ( 1 ) Expanding: xlfalloc.f
-> Loading module (32-bit): /usr/common/usg/cepba2/etc/openmp+mpi32_nsw <-
Loading module in MPI application ...
Module loaded.
-> Searching probes <-
-> Installing probes ( 33 ) <-
-> Activating probes <-
EndProbeCount: 2
-> Starting application <-
. . .
ompttrace: application has been traced
ompttrace: remember to merge intermediate traces by using omp2prv tool:
    omp2prv *.mpit -s *.sym -o poisson242434.prv
```

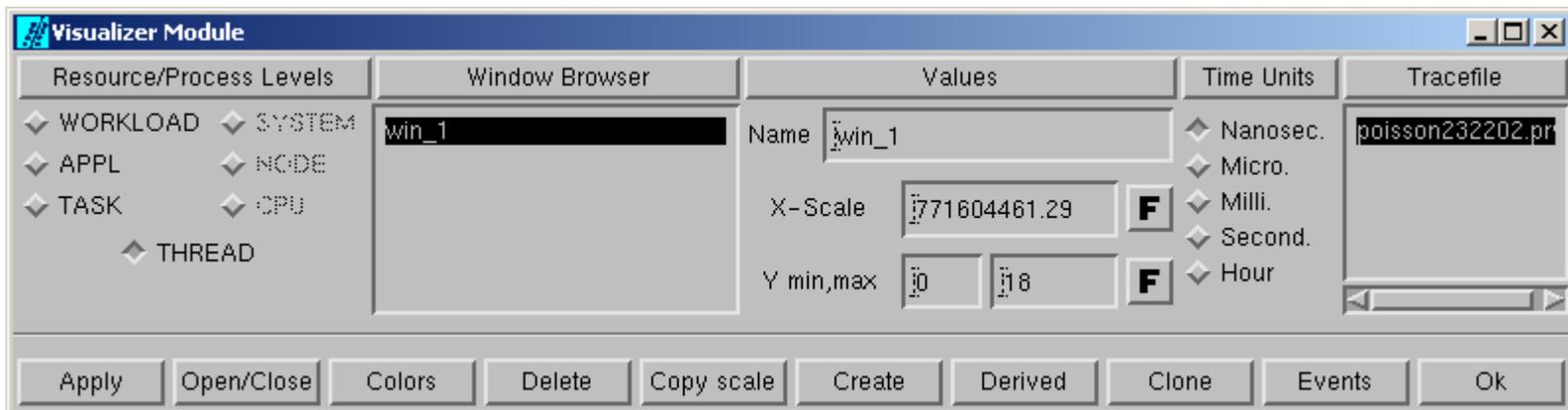
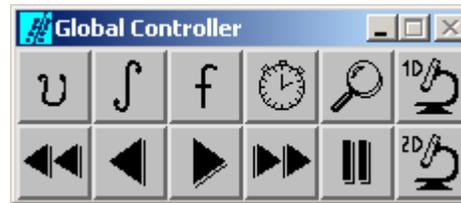
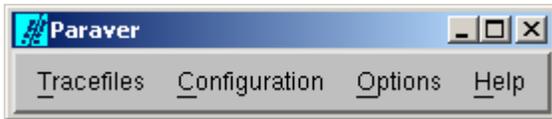

paraver

- Paraver is a powerful tool to visualize states and events in tracefiles
- Each user thread has a trace, the values of states (or events) in several thread traces can be merged into a trace for an MPI task
- MPI tasks can be merged into an application



paraver

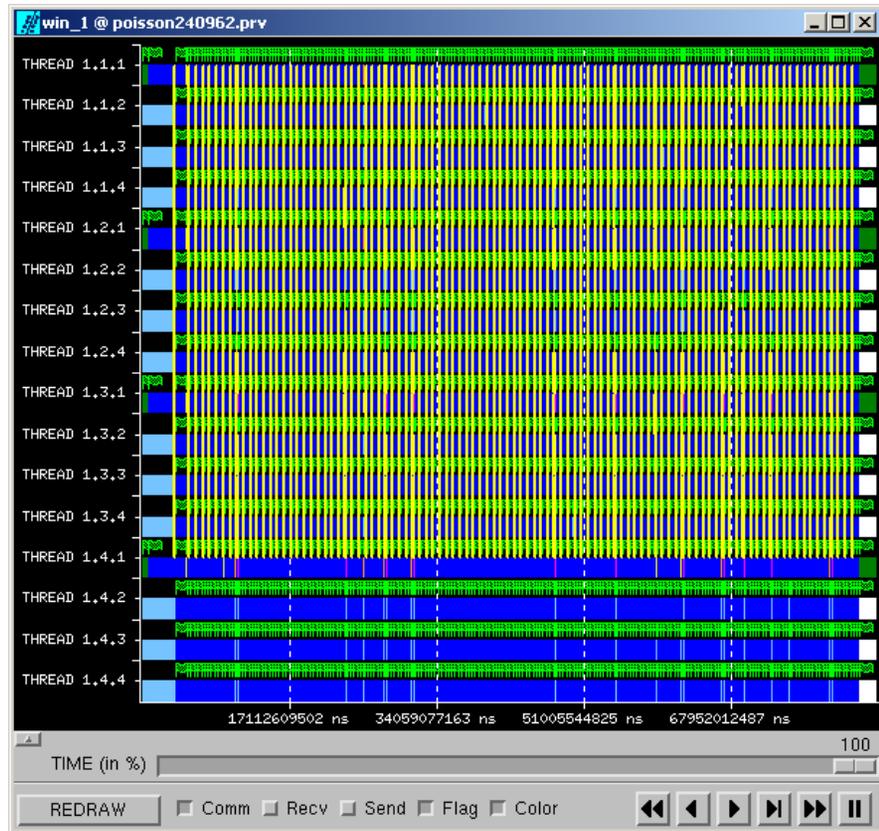
Three windows (and the display window) are created.



paraver

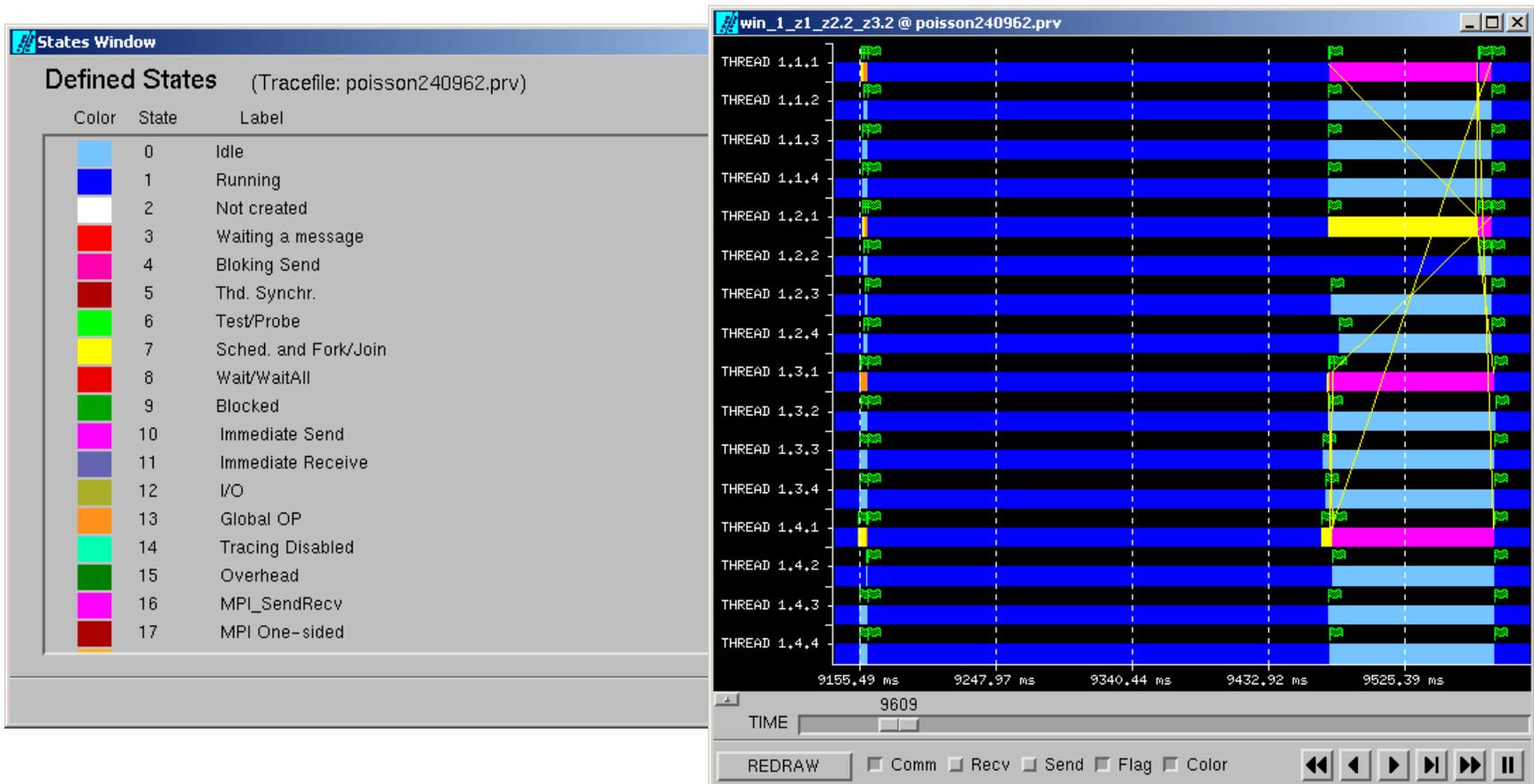
The default display window is usually too cluttered to make out any detail.

Zoom in by right clicking and choose "Zoom" from the menu. Or click the magnifying glass in the Global Controller window. Two additional clicks define the area to be displayed.

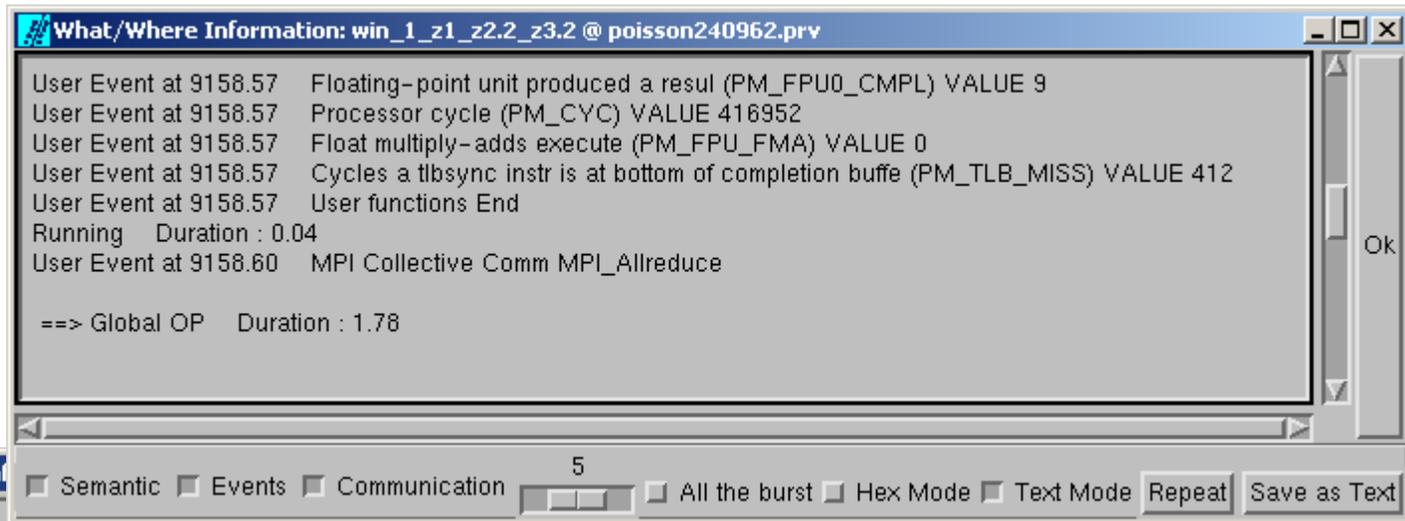


paraver

From "paraver" window, Use "Tracefiles->Trace Information->Defined States window" to bring up a window to show the state to color mapping.



paraver

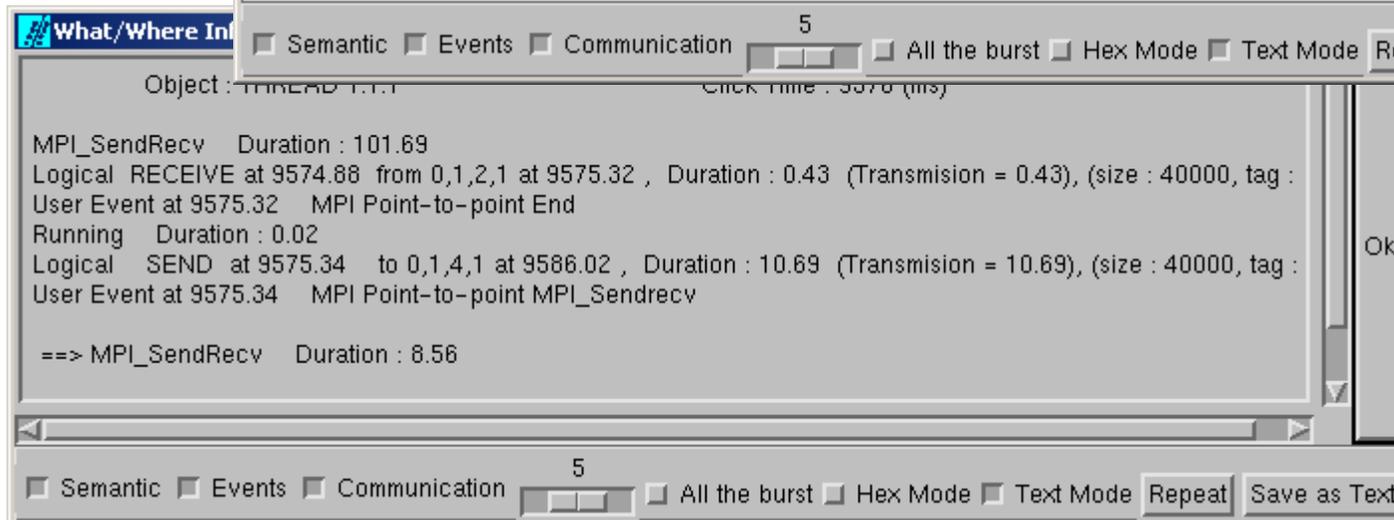


What/Where Information: win_1_z1_z2.2_z3.2 @ poisson240962.prv

```
User Event at 9158.57 Floating-point unit produced a result (PM_FPU0_CMPL) VALUE 9
User Event at 9158.57 Processor cycle (PM_CYC) VALUE 416952
User Event at 9158.57 Float multiply-adds execute (PM_FPU_FMA) VALUE 0
User Event at 9158.57 Cycles a tlbsync instr is at bottom of completion buffe (PM_TLB_MISS) VALUE 412
User Event at 9158.57 User functions End
Running Duration : 0.04
User Event at 9158.60 MPI Collective Comm MPI_Allreduce

==> Global OP Duration : 1.78
```

Ok



What/Where Information: win_1_z1_z2.2_z3.2 @ poisson240962.prv

Object: THREAD 1.1 Click time: 3576 (ms)

Semantic Events Communication All the burst Hex Mode Text Mode Repeat Save as Text

```
MPI_SendRecv Duration : 101.69
Logical RECEIVE at 9574.88 from 0,1,2,1 at 9575.32 , Duration : 0.43 (Transmission = 0.43), (size : 40000, tag :
User Event at 9575.32 MPI Point-to-point End
Running Duration : 0.02
Logical SEND at 9575.34 to 0,1,4,1 at 9586.02 , Duration : 10.69 (Transmission = 10.69), (size : 40000, tag :
User Event at 9575.34 MPI Point-to-point MPI_Sendrecv

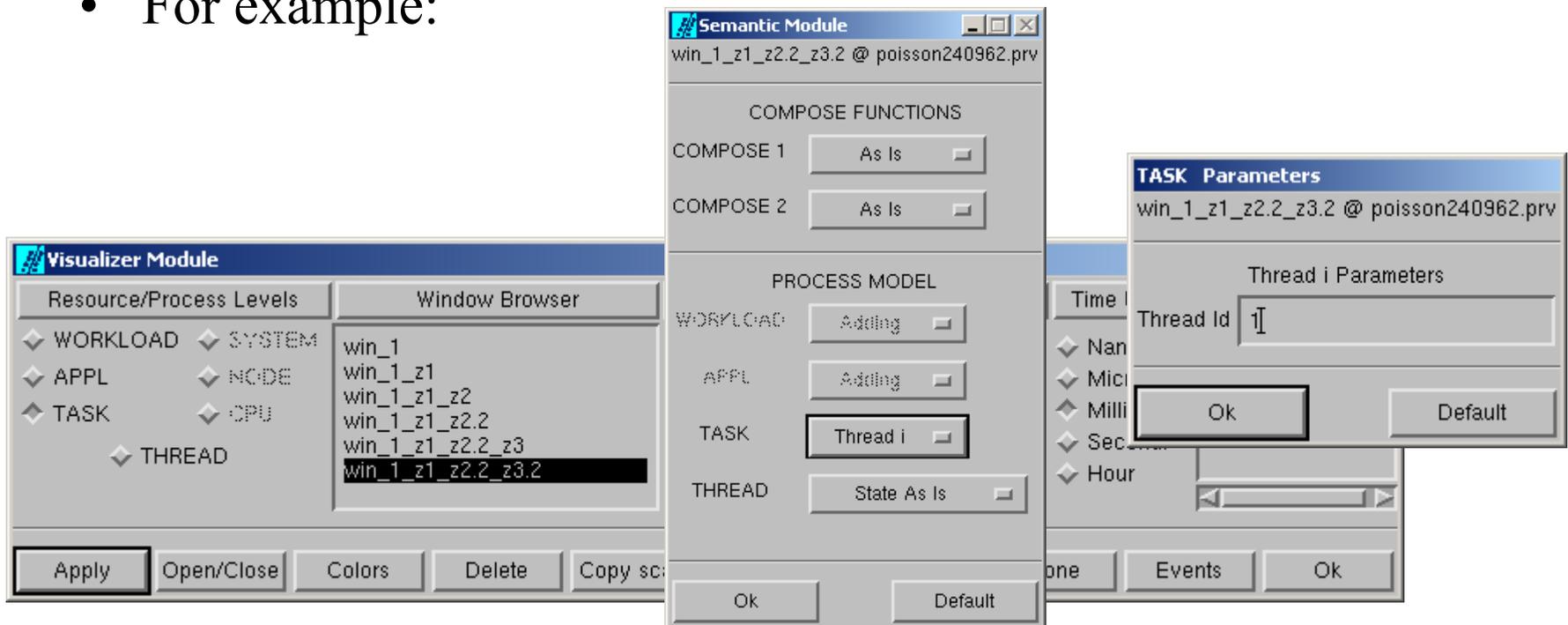
==> MPI_SendRecv Duration : 8.56
```

Ok

Semantic Events Communication All the burst Hex Mode Text Mode Repeat Save as Text

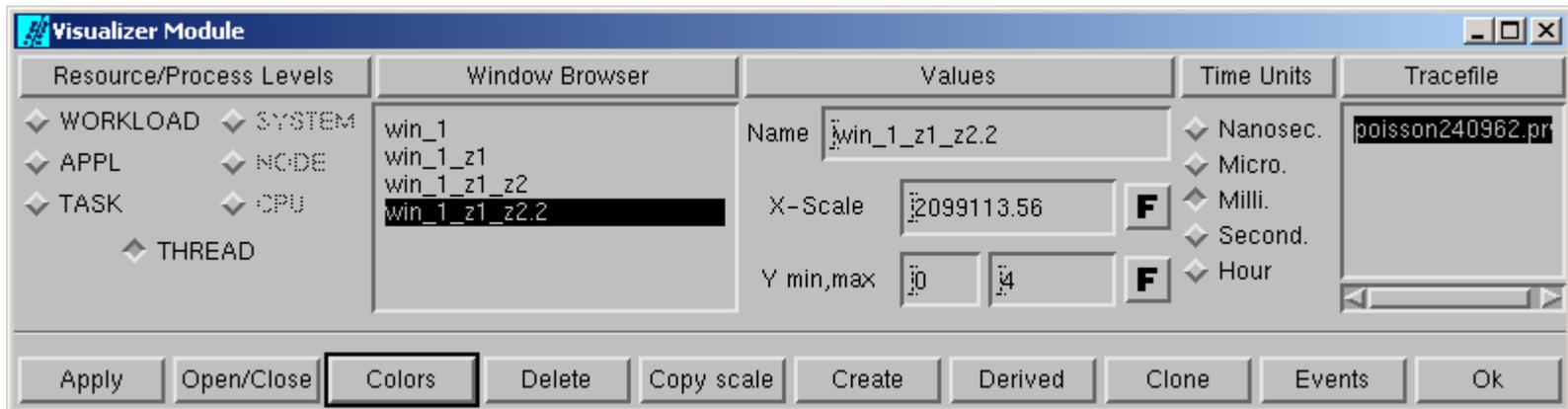
paraver

- Paraver enables you to construct customized displays, but you have to become familiar with the Visualizer, Semantic and Filter modules.
- For example:

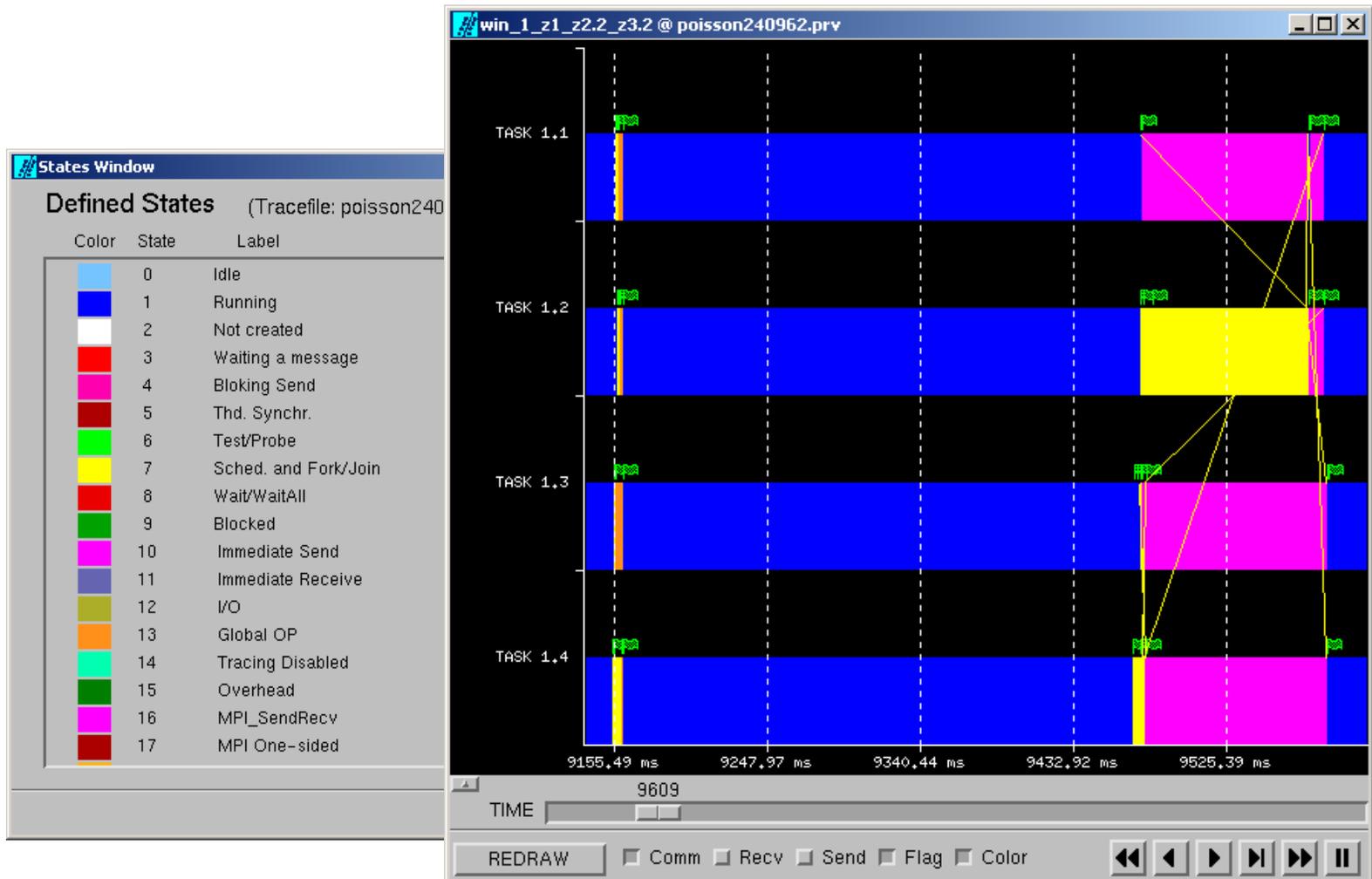


paraver

- If you graph something new, click on **F** in the Visualizer window, and then `Apply` to redraw with the correct scale.

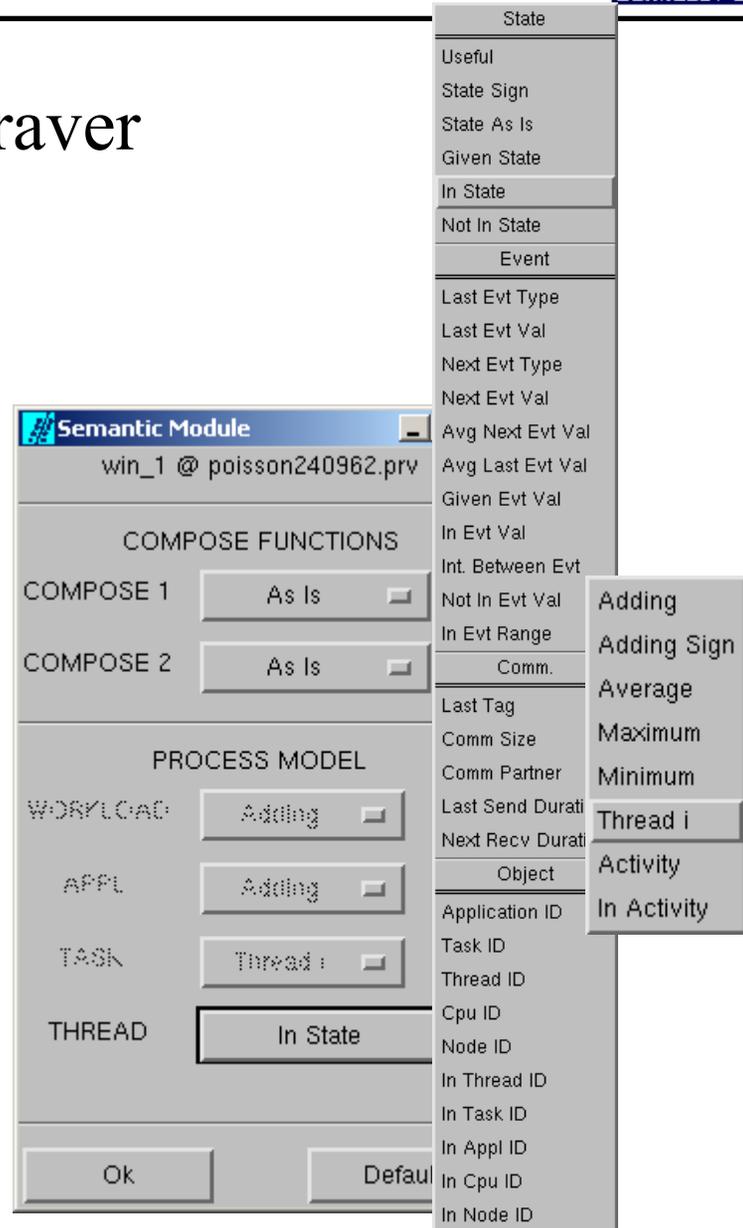


paraver



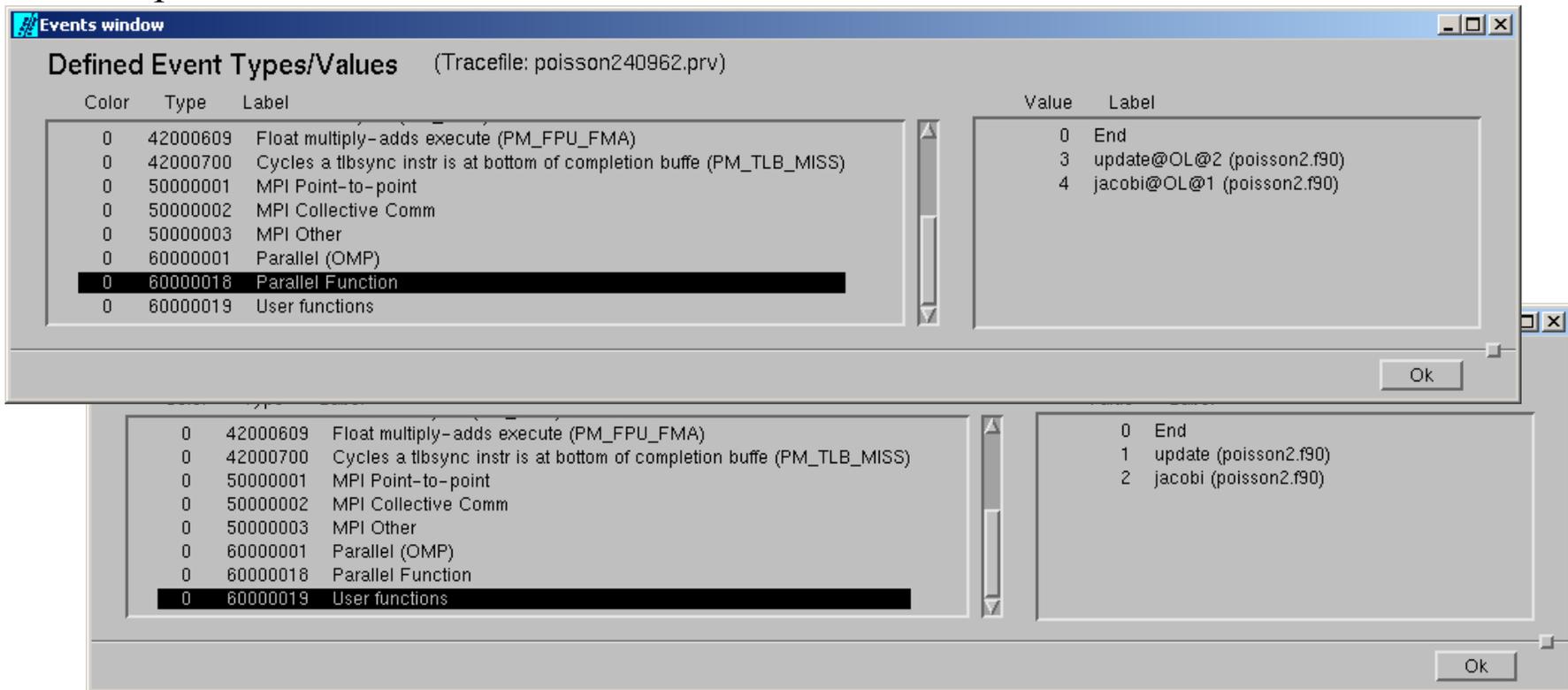
paraver

- Semantic Module has many options to
 - present State Values, Event Types, and Event States to the Visualizer Module
 - merge these values going from threads to tasks to applications



paraver

- Use Paraver window "Tracefiles->Trace Information->Defined Event Type/Values Window" to see all events recorded in the tracefile.
- Suppose we are interested in displaying the two main user functions and the OpenMP versions of those functions

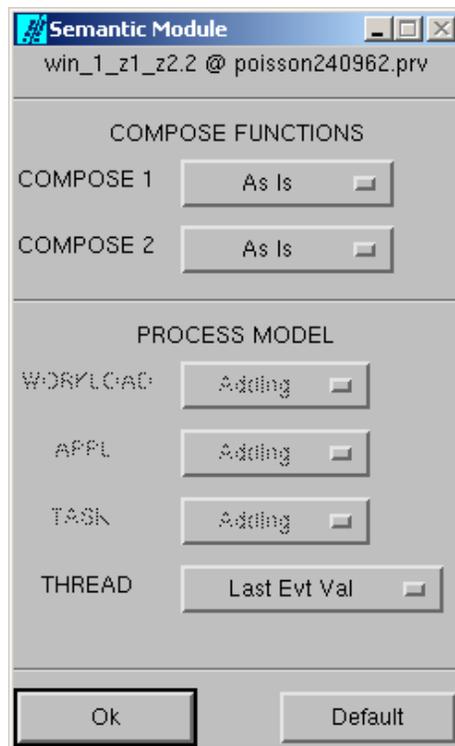


The screenshot shows two instances of the 'Defined Event Types/Values' window. The top window shows the full list of event types, with 'Parallel Function' (type 60000018) selected. The bottom window shows the same list with 'User functions' (type 60000019) selected. Both windows show a list of event types on the left and their corresponding values on the right.

Color	Type	Label	Value	Label
0	42000609	Float multiply-adds execute (PM_FPU_FMA)	0	End
0	42000700	Cycles a tbsync instr is at bottom of completion buffe (PM_TLB_MISS)	3	update@OL@2 (poisson2.f90)
0	50000001	MPI Point-to-point	4	jacobi@OL@1 (poisson2.f90)
0	50000002	MPI Collective Comm		
0	50000003	MPI Other		
0	60000001	Parallel (OMP)		
0	60000018	Parallel Function		
0	60000019	User functions		

paraver

- Use Filter Module to remove all Communication Events, and all User Events not relevant to functions
- Use Semantic Module to display the Function value



paraver

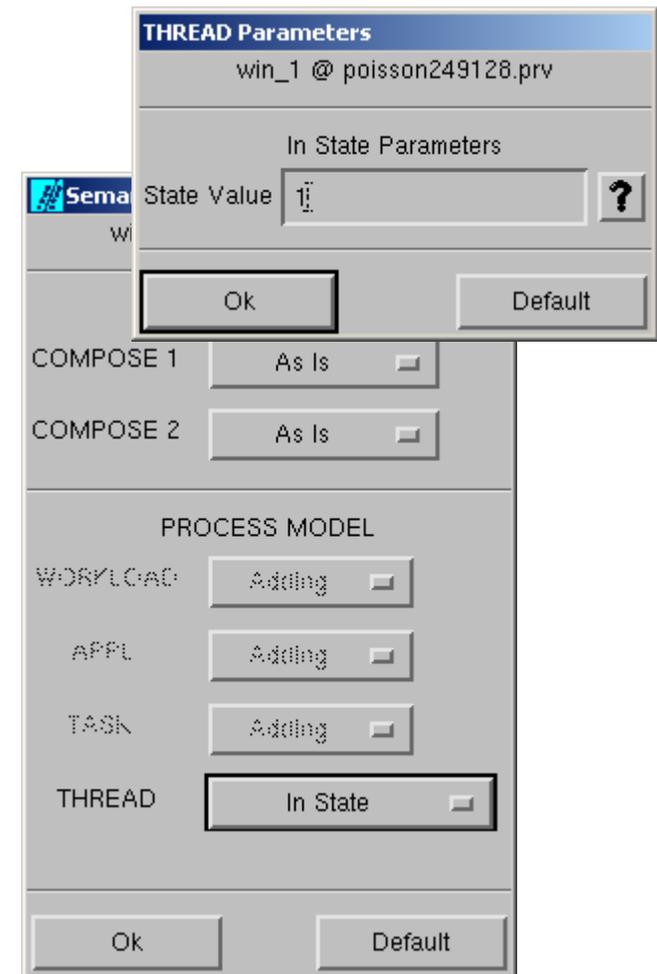


paraver

- The 1D and 2D Analyzer modules can be used to evaluate statistics on state values and event values.
- The 1D Analyzer evaluates a set of functions applied to the output of the semantic module.
- The 2D Analyzer produces a matrix of values, paraver objects vs. states, events, or event values.

1D Analyzer example:

- How much time per thread is spent actually computing (not thread management, not MPI, not idle) ?
- Turn off all filters in Filter Module, set for thread-level Display window in Visualizer Module, use following Semantic Module settings:
- Click on 1D Analyzer button, then click on "All Trace" button in the new window



1D Analyzer

Analyzer: Analysis computed for win_1 @ poisson240962.prv

Row	Avg Semantic Val	# Sends	# Receives	# Events
THREAD 1.1.1	0.93	200	200	8211
THREAD 1.1.2	0.90	0	0	3600
THREAD 1.1.3	0.90	0	0	3600
THREAD 1.1.4	0.90	0	0	3600
THREAD 1.2.1	0.93	200	200	8211
THREAD 1.2.2	0.90	0	0	3600
THREAD 1.2.3	0.90	0	0	3600
THREAD 1.2.4	0.90	0	0	3600
THREAD 1.3.1	0.93	200	200	8211
THREAD 1.3.2	0.90	0	0	3600
THREAD 1.3.3	0.90	0	0	3600
THREAD 1.3.4	0.90	0	0	3600
THREAD 1.4.1	0.93	200	200	8211
THREAD 1.4.2	0.90	0	0	3600
THREAD 1.4.3	0.90	0	0	3600
THREAD 1.4.4	0.90	0	0	3600
Total	14.49	800	800	76044
Average	0.91	50	50	4752.75
Maximum	0.93	200	200	8211
Minimum	0.90	0	0	3600

Buttons: Ok, Save, All Window, All trace, Analyze, Repeat, Calculate All

Summary: Begin Time : 0.00 ms, End Time : 84732.34 ms, Duration : 84732.34 ms

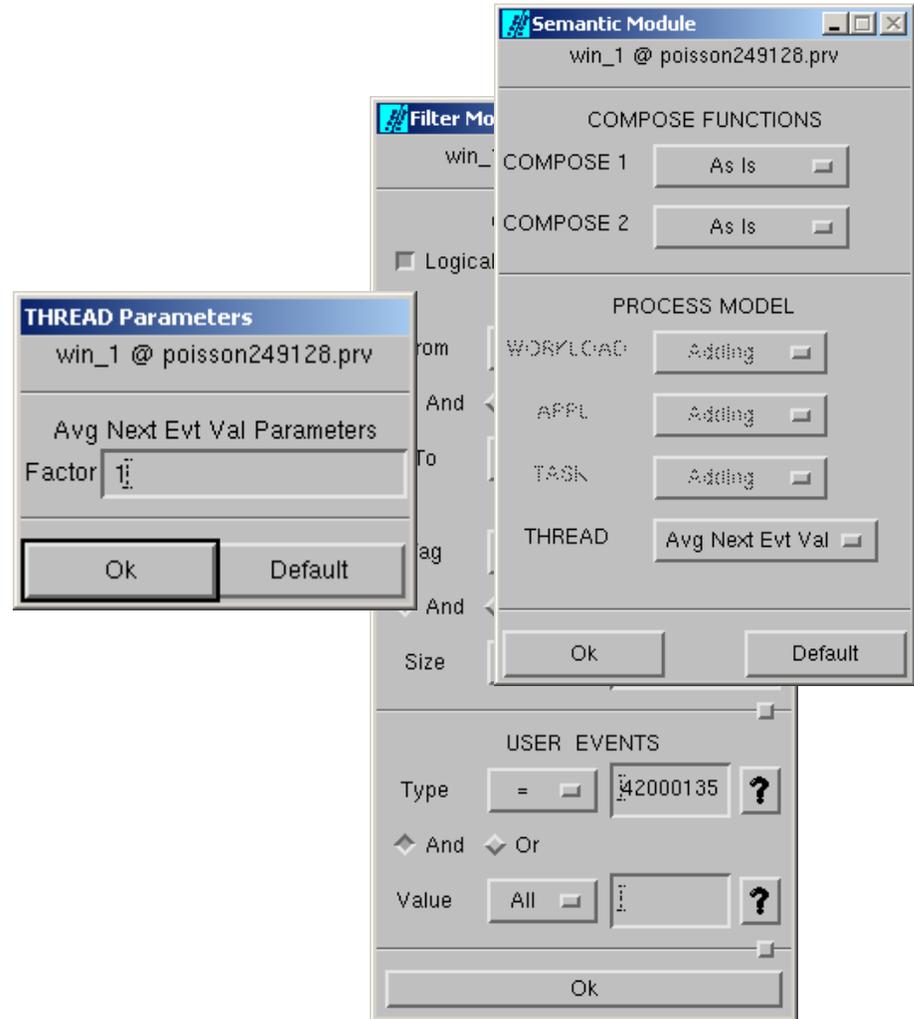
Results show OpenMP threads are about 90% efficient, MPI threads about 93% efficient.

Derived Views

- Power 3 hardware counter values can be visualized over the whole run
- Mflop/s is given approximately by adding the values of the `PM_FPU0_CMPL`, `PM_FPU1_CMPL`, `PM_FPU_FMA` events
- Create three display windows each showing the values of one counter and then add them.

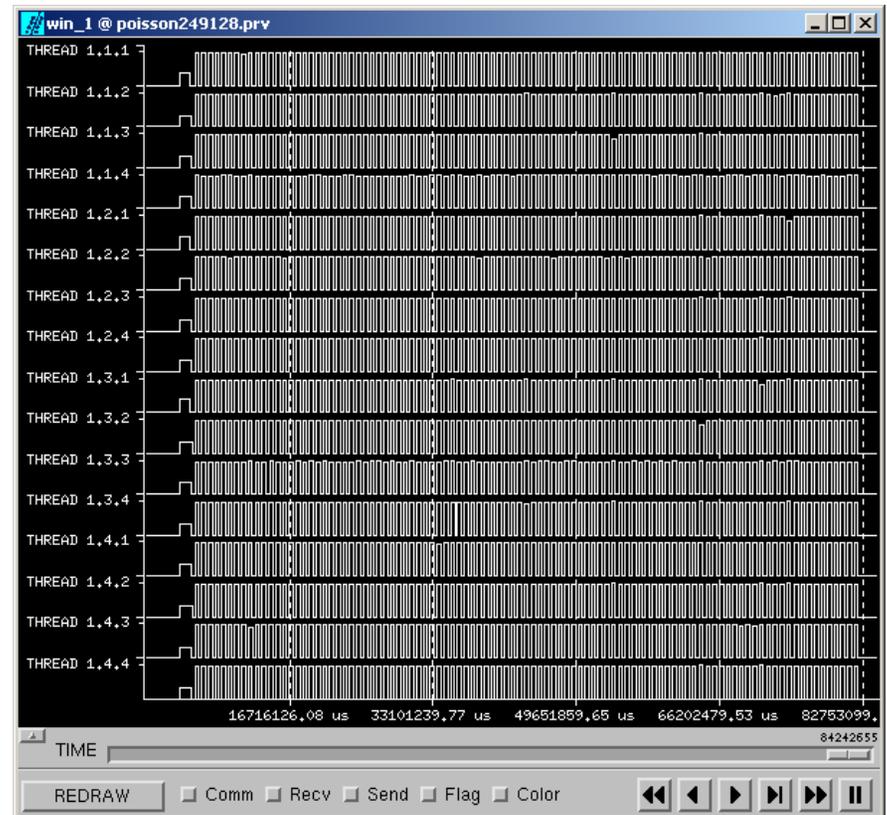
Displaying First Event

- PM_FPU1_CMPL is event 42000135, so filter for this event only
- Choose "Avg Next Evt Val" for the thread Semantic Module
 - Computes last value minus first value divided by time multiplied by "Factor"
 - Units of time are indicated in Visualizer Module



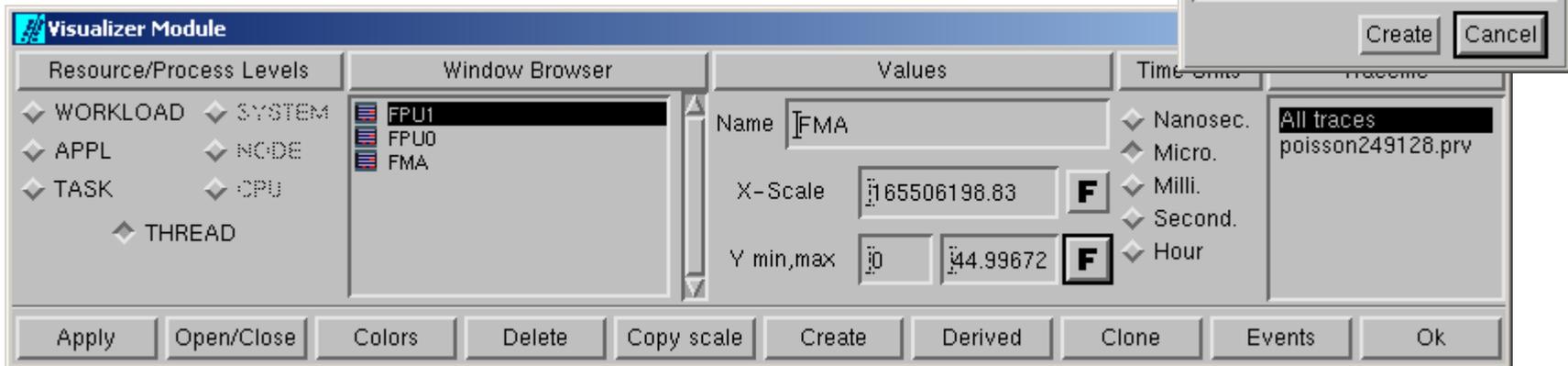
Displaying First Event

- Clicking on the "Color" checkbox plots a graph instead of color coding the value.
- Repeat this procedure with `PM_FPU0_CMPL` and `PM_FPU_FMA`



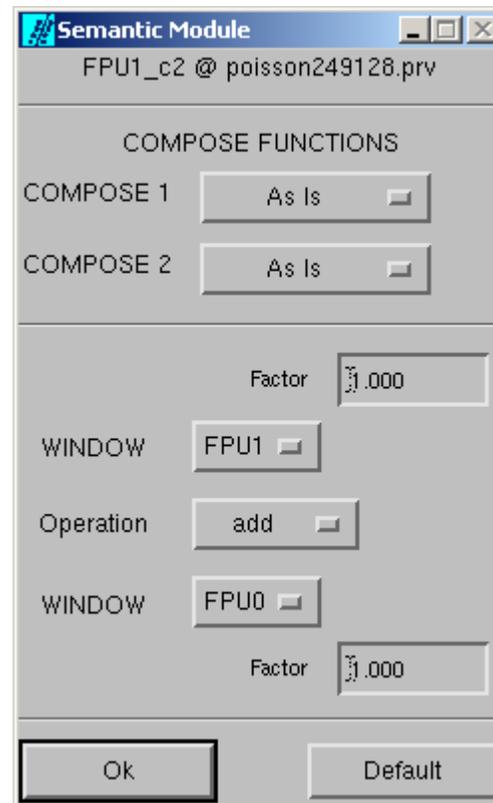
Creating Derived Window

- Visualizer Module should look like this:
 - Three windows have been given a label
- Select a window, then click on "Derived"



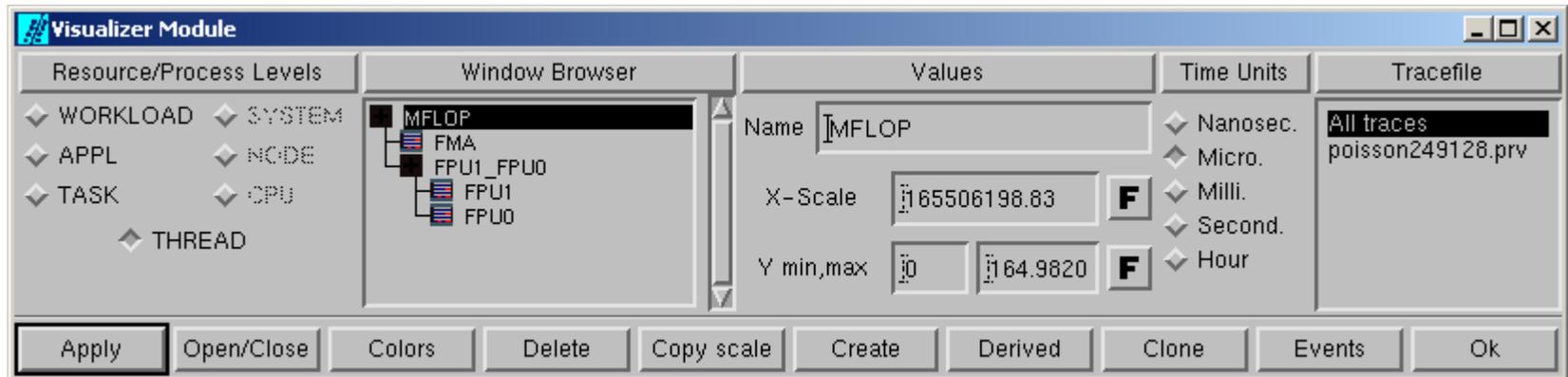
Creating Derived Window

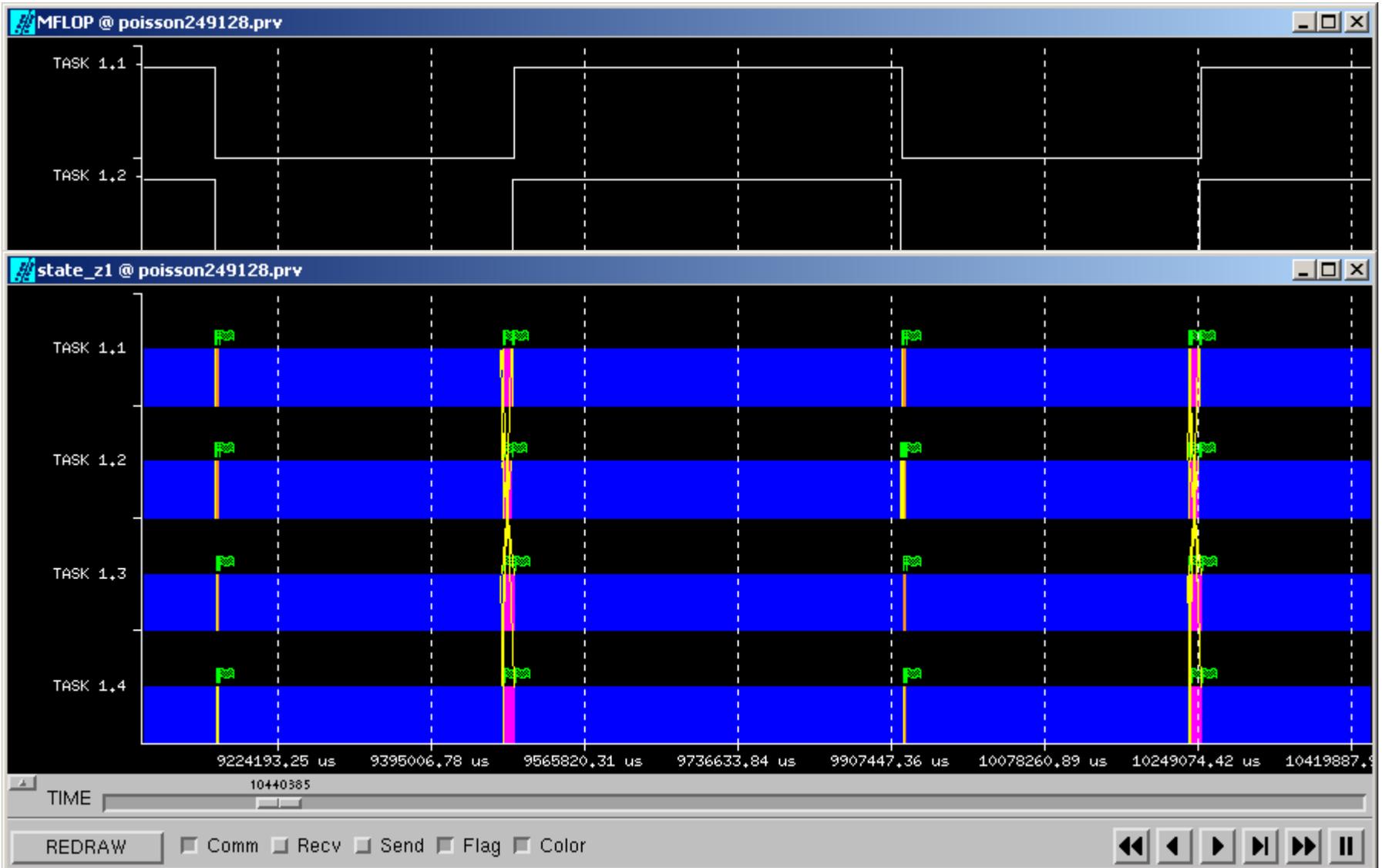
- Semantic Module window changes during creation of a derived view
- Select "add" as the Operation to combine the two windows. Default is "product".



Creating Derived Window

- Add FMA window to FPU0+FPU1 window





Mflop/s 1D Analyzer View

Using the 1D Analyzer on the Mflop window produces the same kind of output as hpmcount.

Analyzer: Analysis computed for win_1_c1_c2 @ poisson249128.prv

Row	Avg Semantic Val	# Sends	# Receives	# Events
THREAD 1.1.1	81.45	400	399	8,204
THREAD 1.1.2	81.45	0	0	3,597
THREAD 1.1.3	81.45	0	0	3,598
THREAD 1.1.4	81.45	0	0	3,597
THREAD 1.2.1	81.46	400	399	8,201
THREAD 1.2.2	81.45	0	0	3,599
THREAD 1.2.3	81.45	0	0	3,600
THREAD 1.2.4	81.45	0	0	3,600
THREAD 1.3.1	81.45	399	399	8,202
THREAD 1.3.2	81.45	0	0	3,598
THREAD 1.3.3	81.45	0	0	3,598
THREAD 1.3.4	81.45	0	0	3,597
THREAD 1.4.1	81.46	398	400	8,199
THREAD 1.4.2	81.45	0	0	3,600
THREAD 1.4.3	81.45	0	0	3,598
THREAD 1.4.4	81.45	0	0	3,599
Total	1,303.22	1,597	1,597	75,987
Average	81.45	100	100	4,749
Maximum	81.46	400	400	8,204
Minimum	81.45	0	0	3,597
Stdev	0.00	173	173	1,993

Buttons: Ok, Save, All Window, All trace, Analyze, Repeat, Calculate All

Summary: Begin Time : 0.00 us, End Time : 84408161.40 us, Duration : 84408161.40 us

Further Reading

- CEPBA Tools
 - <http://www.cepba.upc.es/paraver/index.html>
 - Not completely updated for IBM SP



おわり