
Advanced Tracing Features in Eclipse

Tracing Mini-Summit
LinuxCon 2010
2010-08-09

francois.chouinard@ericsson.com

marc.khouzam@ericsson.com

Overview

› Eclipse Tracing Framework/LTTng

- Introduction
- LTTng Eclipse Integration
- Perspective and Views
- Upcoming Features

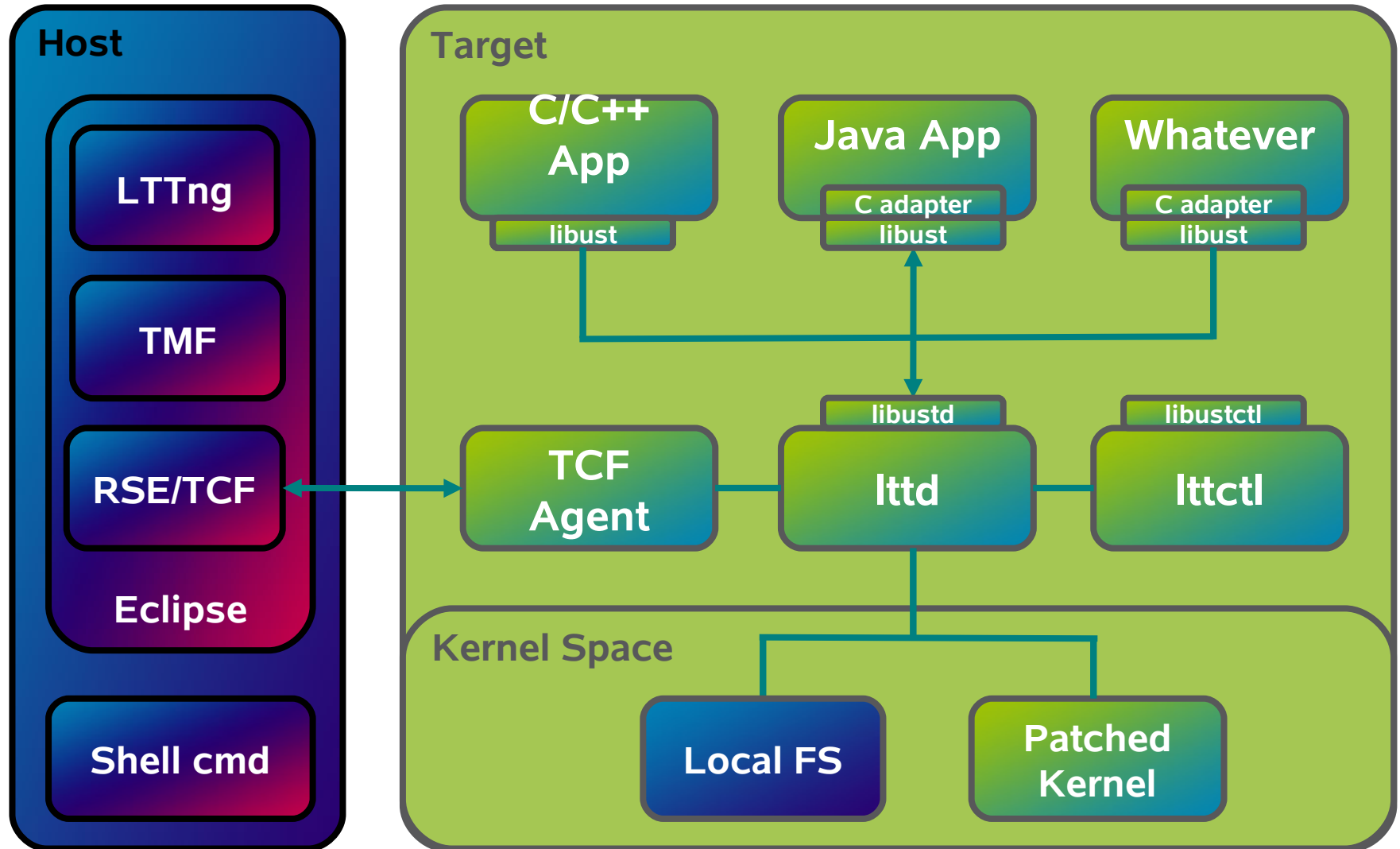
› Eclipse GDB Tracepoints

- Dynamic Tracepoints
- Data Visualization
- Static Tracepoints
- Planned Features

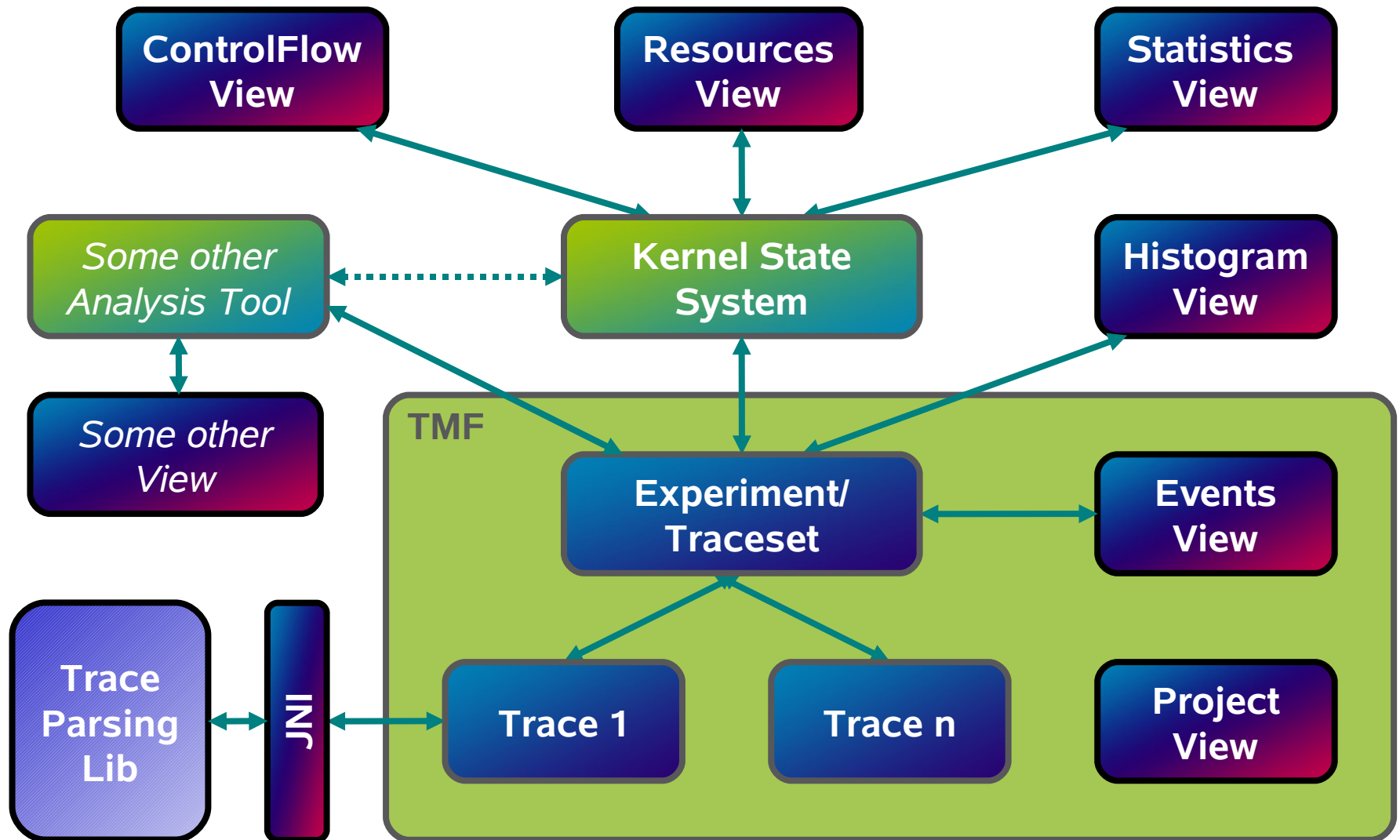
LTTng – Introduction

- › Making use of LTTng with LTTv
- › Integration of different tools in Eclipse
- › Focus on the new LTTng integration in Eclipse

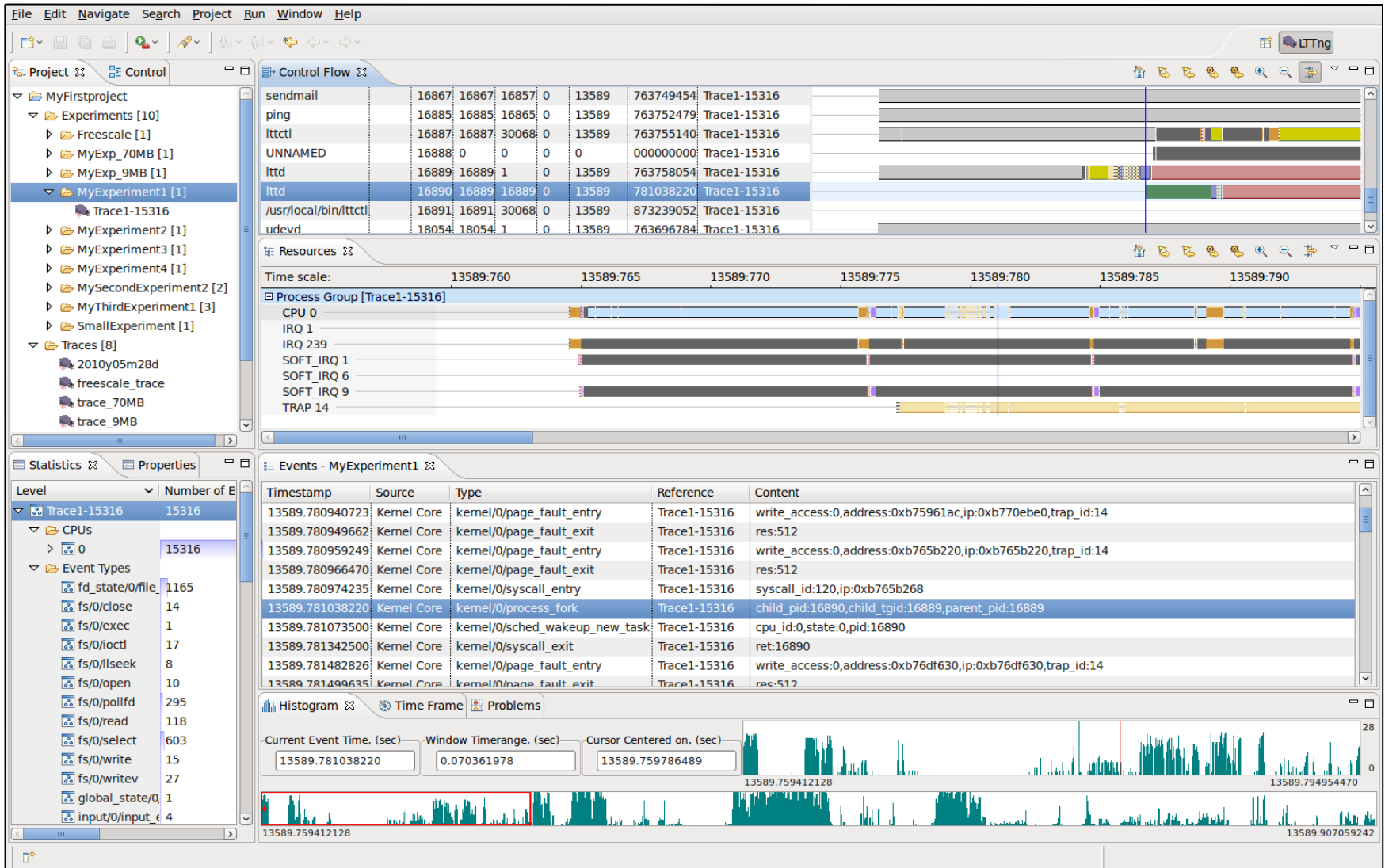
LTTng Eclipse Integration



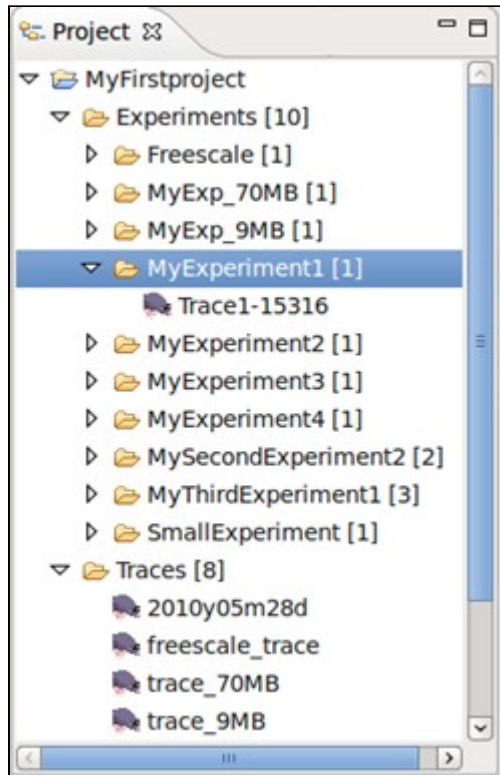
LTTng Eclipse Architecture



LTTng Perspective



LTTng – Project View



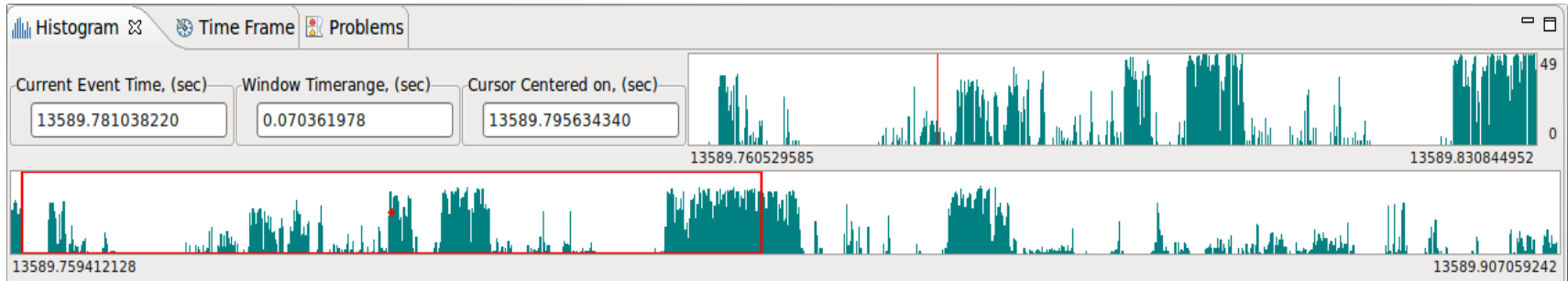
- › Projects are used to group traces that you wish to correlate
- › Experiments are specific correlations between selected trace files
- › Traces are all trace files currently included in the project

LTTng – Events View

Events - MyExperiment1				
Timestamp	Source	Type	Reference	Content
13589.780940723	Kernel Core	kernel/0/page_fault_entry	Trace1-15316	write_access:0,address:0xb75961ac,ip:0xb770ebe0,trap_id:14
13589.780949662	Kernel Core	kernel/0/page_fault_exit	Trace1-15316	res:512
13589.780959249	Kernel Core	kernel/0/page_fault_entry	Trace1-15316	write_access:0,address:0xb765b220,ip:0xb765b220,trap_id:14
13589.780966470	Kernel Core	kernel/0/page_fault_exit	Trace1-15316	res:512
13589.780974235	Kernel Core	kernel/0/syscall_entry	Trace1-15316	syscall_id:120,ip:0xb765b268
13589.781038220	Kernel Core	kernel/0/process_fork	Trace1-15316	child_pid:16890,child_tgid:16889,parent_pid:16889
13589.781073500	Kernel Core	kernel/0/sched_wakeup_new_task	Trace1-15316	cpu_id:0,state:0,pid:16890
13589.781342500	Kernel Core	kernel/0/syscall_exit	Trace1-15316	ret:16890
13589.781482826	Kernel Core	kernel/0/page_fault_entry	Trace1-15316	write_access:0,address:0xb76df630,ip:0xb76df630,trap_id:14
13589.781499635	Kernel Core	kernel/0/page_fault_exit	Trace1-15316	res:512
13589.781513465	Kernel Core	kernel/0/syscall_entry	Trace1-15316	syscall_id:240,ip:0xb7705416
13589.781552921	Kernel Core	kernel/0/sched_schedule	Trace1-15316	prev_pid:16889,next_pid:16888,prev_state:1
13589.781772737	Kernel Core	kernel/0/sched_schedule	Trace1-15316	prev_pid:16888,next_pid:16887,prev_state:64
13589.781953709	Kernel Core	mm/0/page_free	Trace1-15316	order:1,pfn:79400
13589.782017603	Kernel Core	fd_state/0/file_descriptor	Trace1-15316	fd:0,filename:/dev/null,pid:1883
13589.782022532	Kernel Core	fd_state/0/file_descriptor	Trace1-15316	fd:1,filename:/dev/null,pid:1883

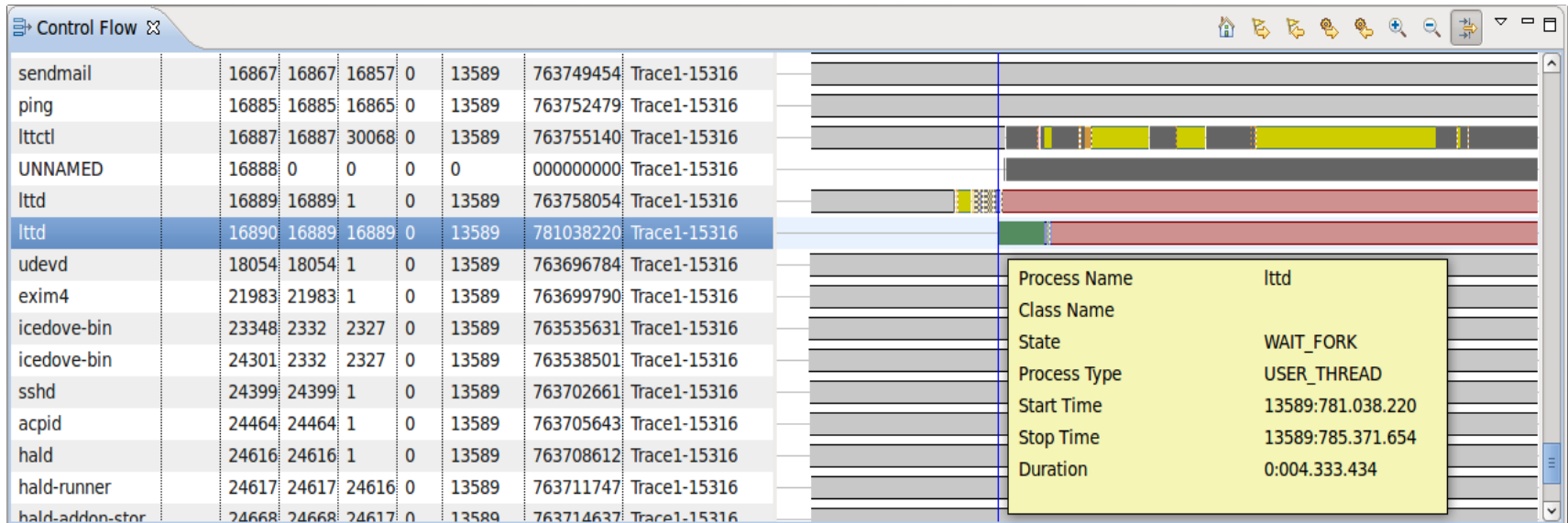
- › 'Raw' merged events in chronological order
- › Synchronized on timestamp with other views
- › Upcoming feature:
 - Event filtering on time range, event type, field value (e.g. pid), ...
 - Individual trace tabs

LTTng – Histogram View



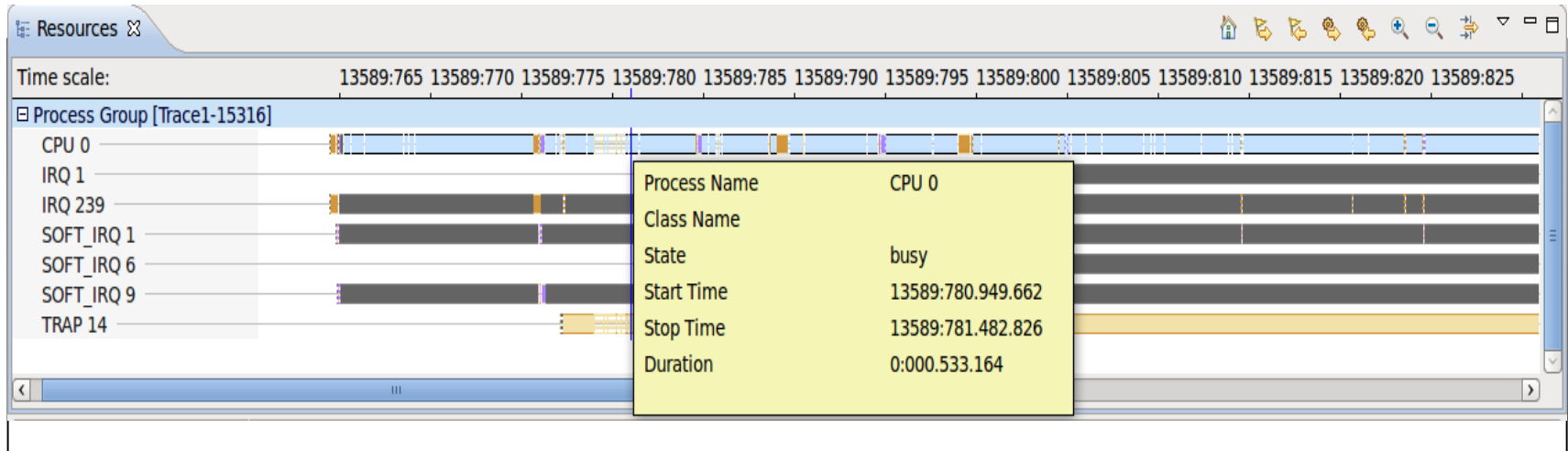
- › Event distribution over full traceset and selected window
- › Controls to modify current event and event window
- › Synchronized on current window and current event
- › Upcoming feature:
 - Zooming the selected window using the mouse

LTTng – Control Flow View



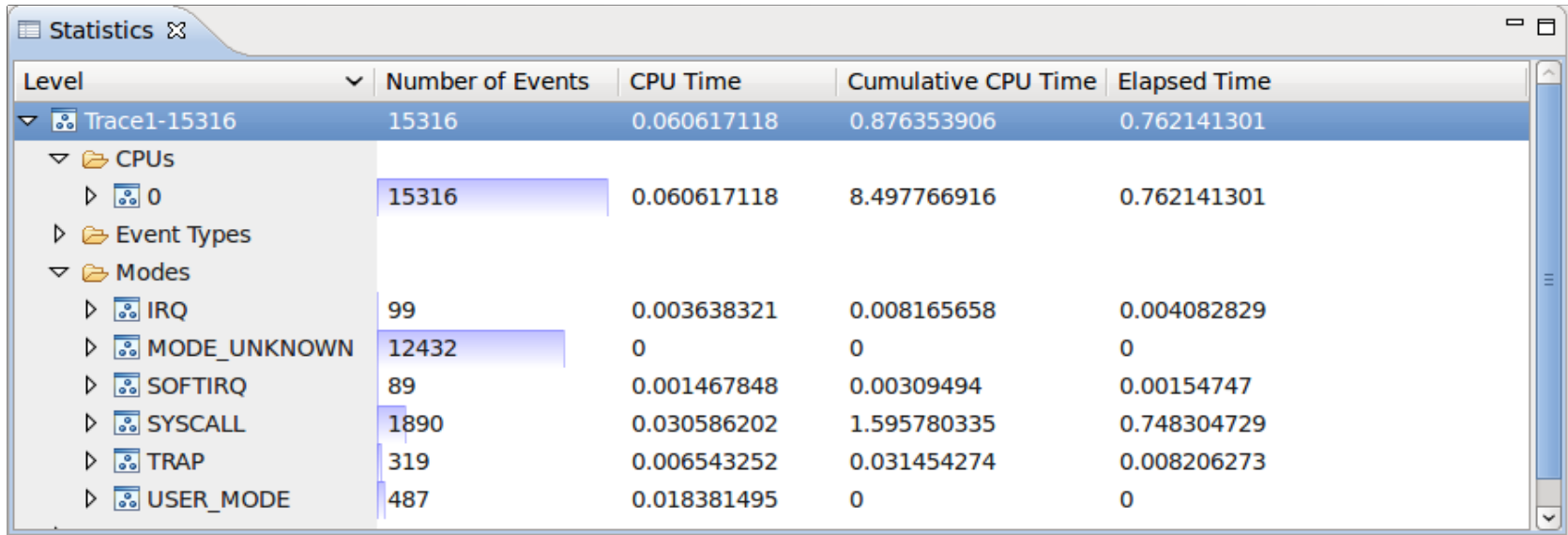
- › Displays processes states (color-coded) over time
- › State 'tooltips' through hovering
- › Zooming and filtering
- › Quick navigation between processes, states
- › Upcoming features :
 - Color legend
 - Configurable color scheme

LTTng – Resources View



- › Displays system resource states (color-coded) over time
- › State 'tooltips'
- › Zooming and filtering
- › Quick navigation between resources, states
- › Upcoming features :
 - Color legend
 - Configurable color scheme

LTTng – Statistics View



The screenshot shows the 'Statistics' window in LTTng. It displays a hierarchical tree of event statistics. The tree is expanded to show the 'Modes' section, which includes 'IRQ', 'MODE_UNKNOWN', 'SOFTIRQ', 'SYSCALL', 'TRAP', and 'USER_MODE'. Each mode has associated statistics for 'Number of Events', 'CPU Time', 'Cumulative CPU Time', and 'Elapsed Time'.

Level	Number of Events	CPU Time	Cumulative CPU Time	Elapsed Time
Trace1-15316	15316	0.060617118	0.876353906	0.762141301
CPUs				
0	15316	0.060617118	8.497766916	0.762141301
Event Types				
Modes				
IRQ	99	0.003638321	0.008165658	0.004082829
MODE_UNKNOWN	12432	0	0	0
SOFTIRQ	89	0.001467848	0.00309494	0.00154747
SYSCALL	1890	0.030586202	1.595780335	0.748304729
TRAP	319	0.006543252	0.031454274	0.008206273
USER_MODE	487	0.018381495	0	0

- › Displays basic CPU usage statistics
- › Upcoming feature:
 - Make the view generic (decoupled from the kernel events structure)

LTTng – Upcoming Features

› General

- Tracing tool control
- Trace streaming
- Correlation of heterogeneous traces
- User Space Tracing
- GDB Tracepoints
- Source lookup
- Performance tuning

› Analyses

- Time correction (traces synchronization)
 - › Multi-core, multi-level, multi-node
- Timing dependencies (processes interactions e.g. startup time)
- Pattern matching (security e.g. intrusion detection)

LTTng – Pointers

LTTng Eclipse Project (<http://www.eclipse.org/linuxtools/projectPages/lttng>)

LTTng Eclipse Wiki (http://wiki.eclipse.org/Linux_Tools_Project/LTTng)

Linux Tools (<http://www.eclipse.org/linuxtools/index.php>)

Update Site (<http://download.eclipse.org/technology/linuxtools/update>)

LTTng Project (<http://lttng.org>)

Tracing Wiki (<http://lttng.org/tracingwiki/index.php/TracingBook>)

Dynamic Tracing

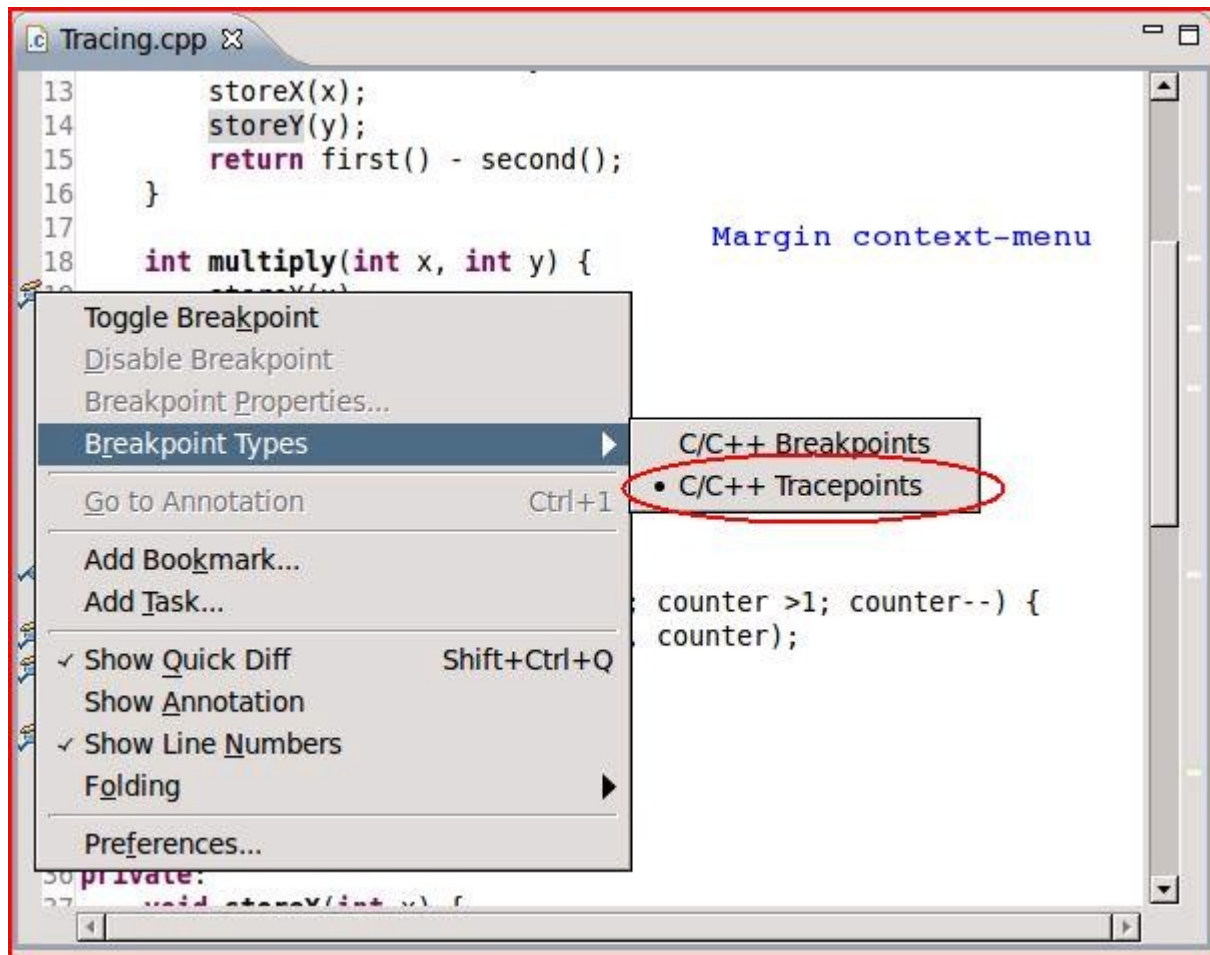
- › What if existing traces don't give info needed?
- › What about systems that are not instrumented?
- GDB's Dynamic Tracepoints
- Integration within Eclipse

Eclipse Tracepoints

- › Creation of tracepoint as is done as for breakpoints
- › Enable/Disable
- › Dynamic condition
- › Specification of data to be gathered using symbolic expressions and memory addresses (actions)
- › Pass count
- › Trace-state variables can be used in conditions and actions

Eclipse Tracepoints Selection

- › Tracepoints treated as breakpoints



Eclipse Tracepoints Display

- › Tracepoints
- › Tracepoints with actions

The screenshot displays the Eclipse IDE interface. On the left, the 'Tracing.cpp' editor shows C++ code with several tracepoints (bug icons) placed at lines 19, 29, 30, and 32. On the right, the 'Breakpoints' view lists the configured tracepoints, each with a checked checkbox and a bug icon. A red rectangle highlights the tracepoints at lines 7, 19, 29, 30, and 32 in both the code editor and the breakpoints list.

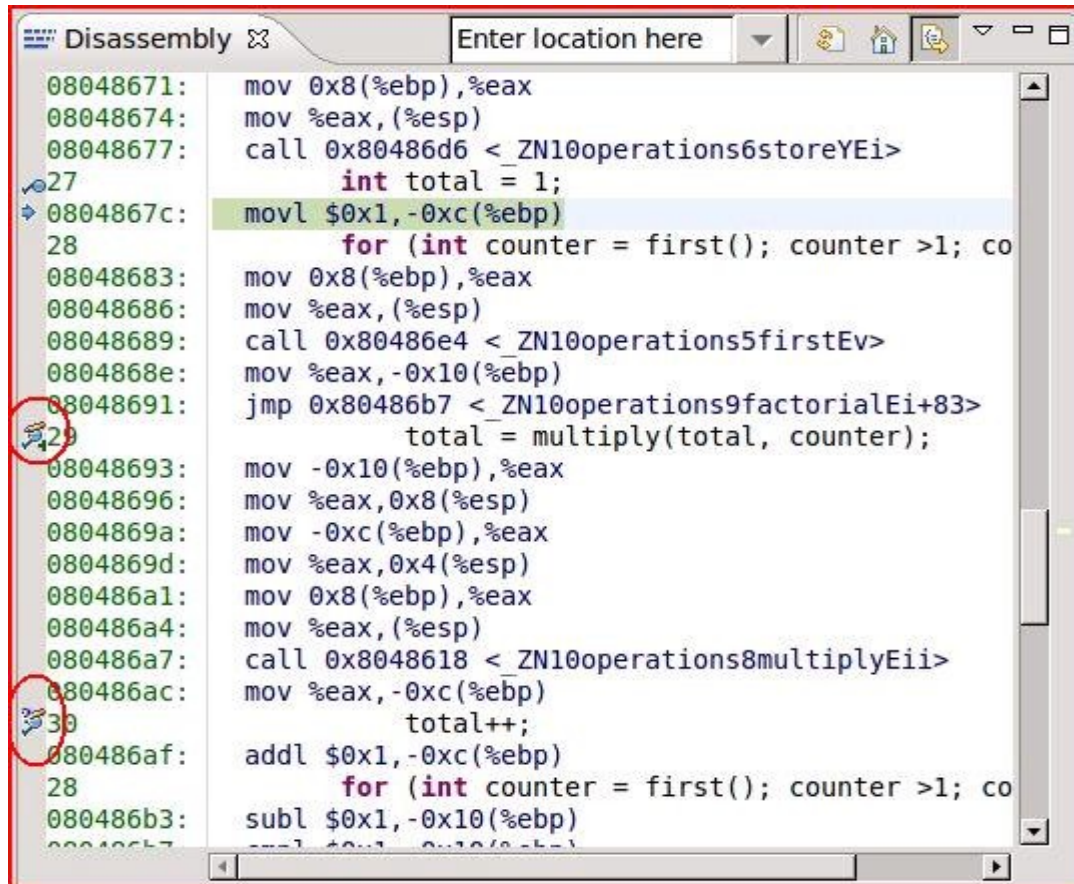
```
Tracing.cpp
13     storeX(x);
14     storeY(y);
15     return first() - second();
16 }
17
18 int multiply(int x, int y) {
19     storeX(x);
20     storeY(y);
21     return first() * second();
22 }
23
24 int factorial(int y) {
25     storeY(y);
26
27     int total = 1;
28     for (int counter = first(); counter > 1; counter--) {
29         total = multiply(total, counter);
30         total++;
31     }
32     return total;
33 }
34
35
```

Breakpoints

- ☒ /home/lmckhou/runtime/Tracing/src/Tracing.cpp [line: 27]
- ☒ /home/lmckhou/runtime/Tracing/src/Tracing.cpp [line: 64]
- ☒ /home/lmckhou/runtime/Tracing/src/Tracing.cpp [line: 7]
- ☒ /home/lmckhou/runtime/Tracing/src/Tracing.cpp [line: 19]
- ☒ /home/lmckhou/runtime/Tracing/src/Tracing.cpp [line: 29]
- ☒ /home/lmckhou/runtime/Tracing/src/Tracing.cpp [line: 30]
- ☒ /home/lmckhou/runtime/Tracing/src/Tracing.cpp [line: 32]

Eclipse Tracepoints Disassembly

- › Disassembly view support for Tracepoints
- › Tracepoint with condition



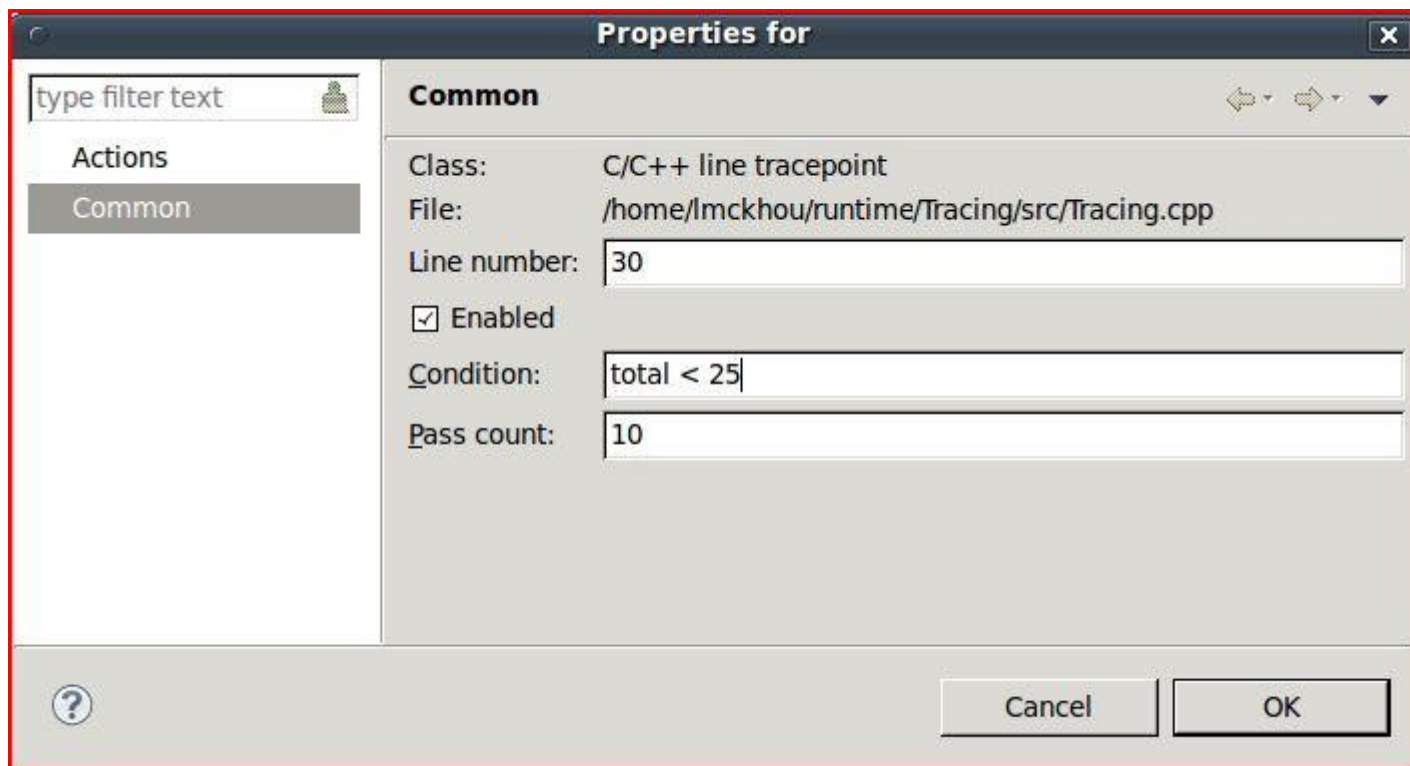
The screenshot shows the Eclipse IDE's Disassembly view. The title bar reads "Disassembly" and "Enter location here". The main area displays assembly code with corresponding C code comments. Two tracepoints are visible, marked with a bug icon and a red circle:

- Tracepoint 27: `int total = 1;` (highlighted in blue)
- Tracepoint 29: `total = multiply(total, counter);`

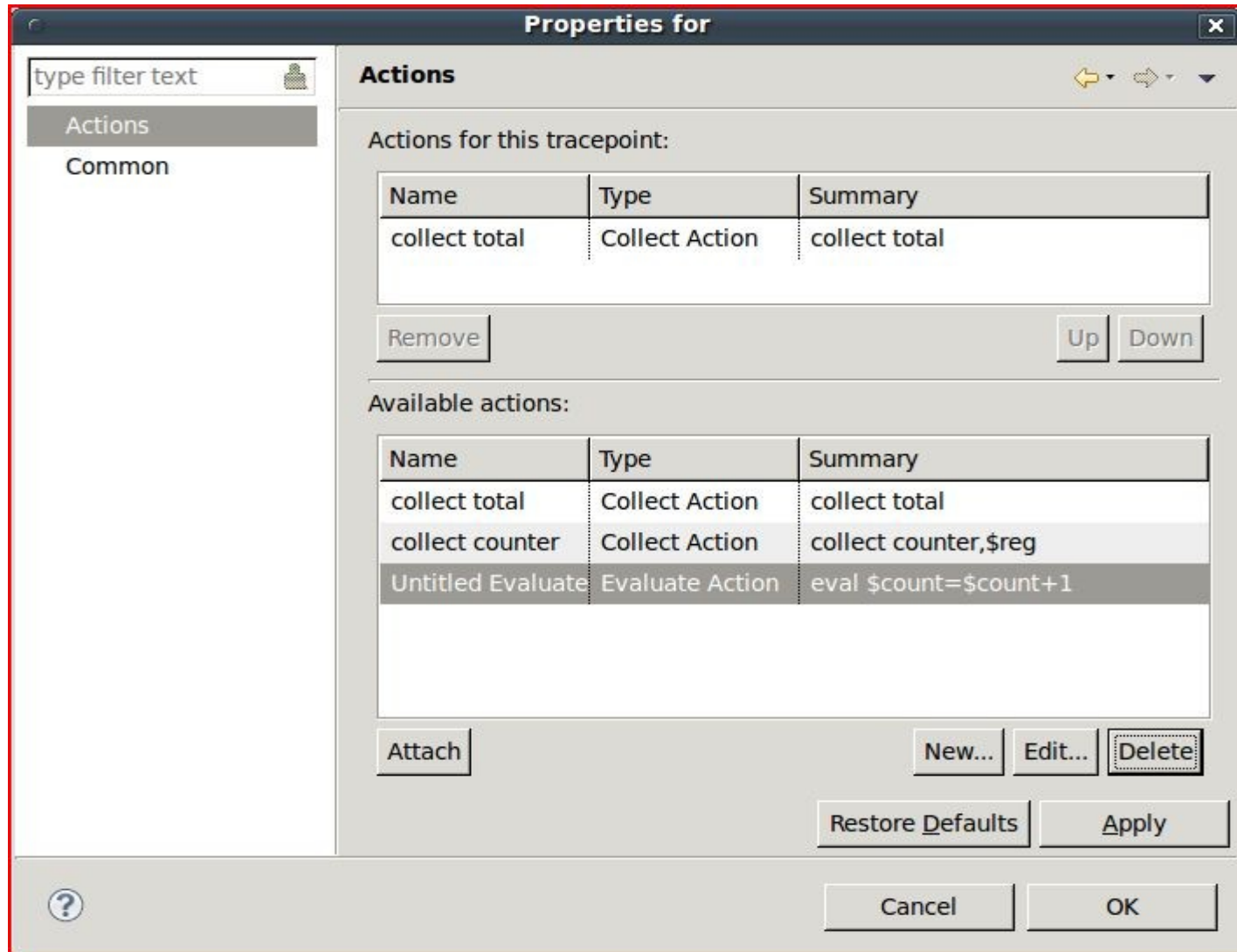
The assembly code includes instructions like `mov 0x8(%ebp),%eax`, `mov %eax,(%esp)`, `call 0x80486d6 <ZN10operations6storeYEi>`, `movl $0x1,-0xc(%ebp)`, `for (int counter = first(); counter >1; co`, `mov 0x8(%ebp),%eax`, `mov %eax,(%esp)`, `call 0x80486e4 <ZN10operations5firstEv>`, `mov %eax,-0x10(%ebp)`, `jmp 0x80486b7 <ZN10operations9factorialEi+83>`, `mov -0x10(%ebp),%eax`, `mov %eax,0x8(%esp)`, `mov -0xc(%ebp),%eax`, `mov %eax,0x4(%esp)`, `mov 0x8(%ebp),%eax`, `mov %eax,(%esp)`, `call 0x8048618 <ZN10operations8multiplyEii>`, `mov %eax,-0xc(%ebp)`, `addl $0x1,-0xc(%ebp)`, `for (int counter = first(); counter >1; co`, and `subl $0x1,-0x10(%ebp)`.

Eclipse Tracepoints Properties

- › Tracepoints properties
 - Location
 - Enablement
 - Condition
 - Pass count

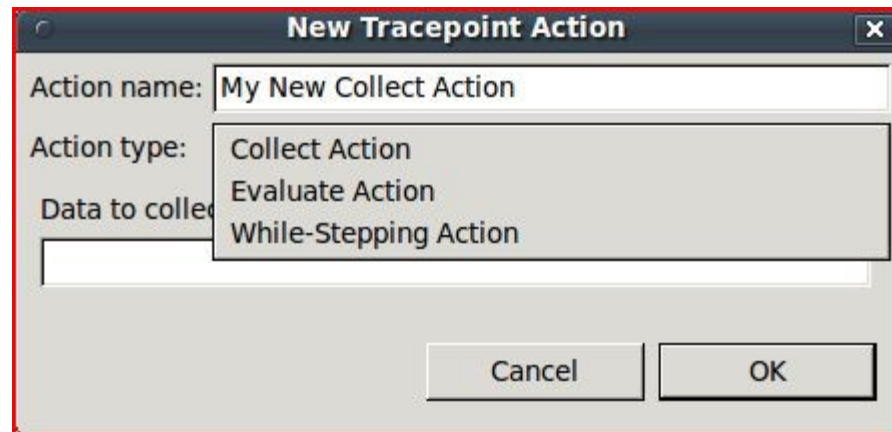


Eclipse Tracepoints Actions

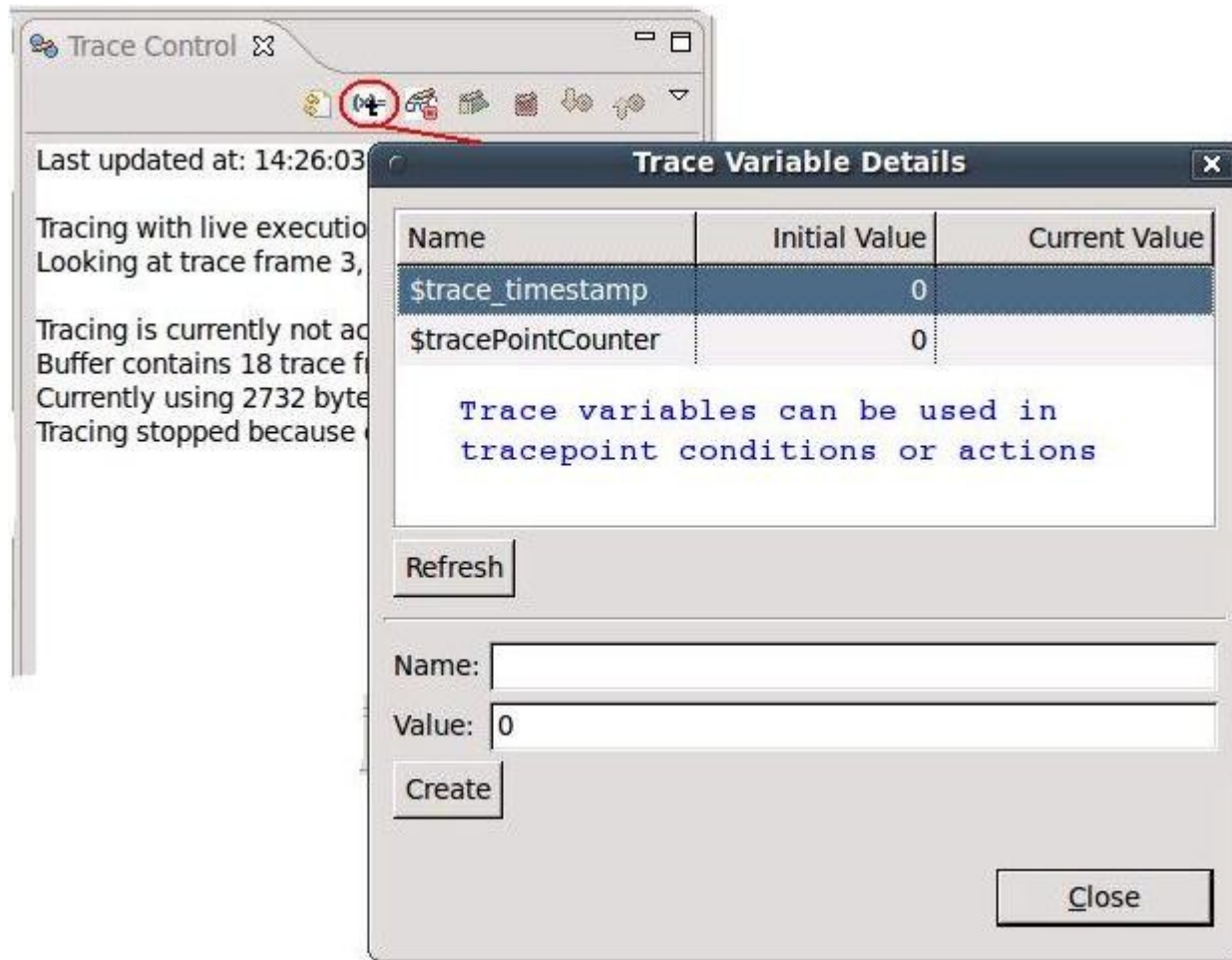


Eclipse Tracepoints Actions

- › Tracepoints action types
 - Collect
 - Evaluate
 - While-Stepping
 - › Collect
 - › Evaluate



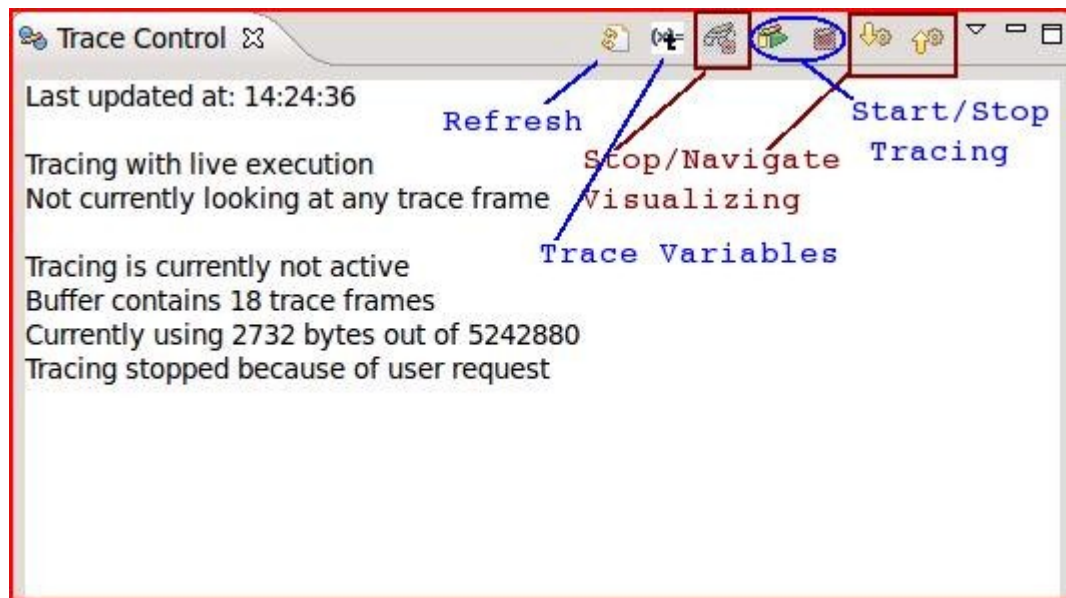
Eclipse Tracepoints Variables



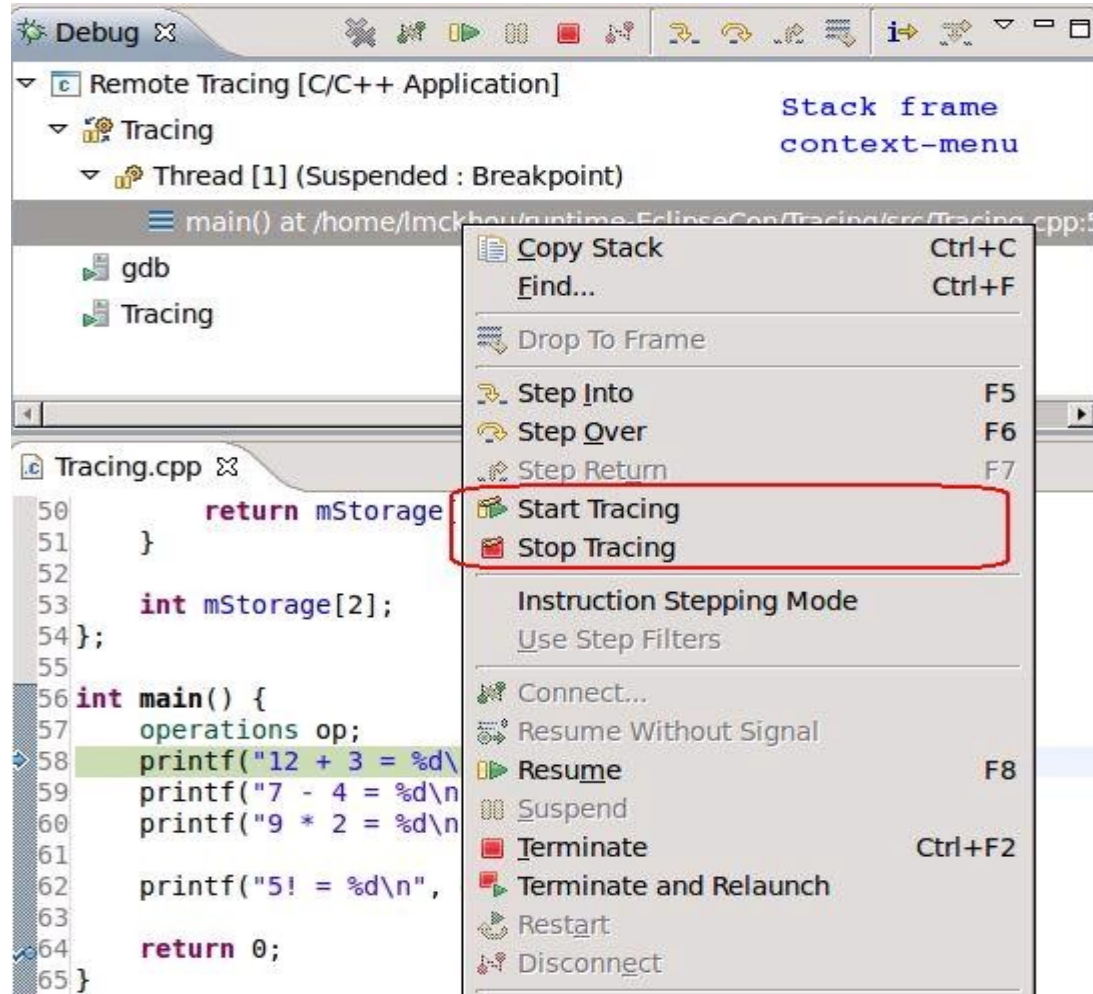
Eclipse Tracepoints Control

› Trace Control View

- Refreshing info
- Trace Variables
- Start/Stop Tracing
- Navigate during Visualization
- Stop Visualization

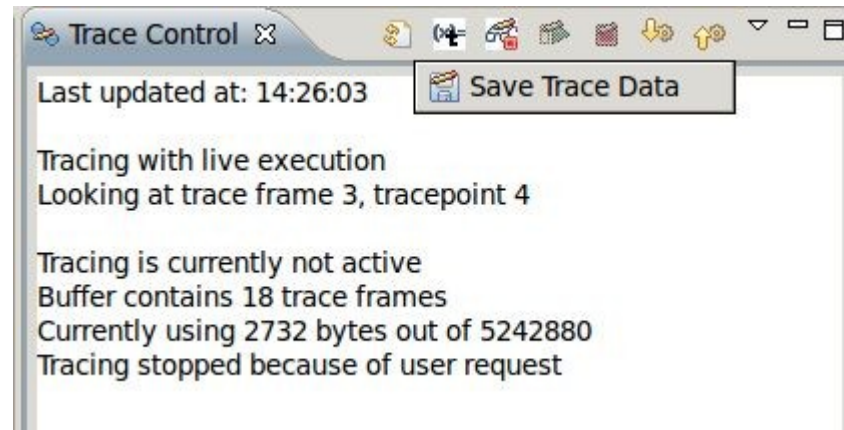


Eclipse Tracepoints Control



Eclipse Trace Data

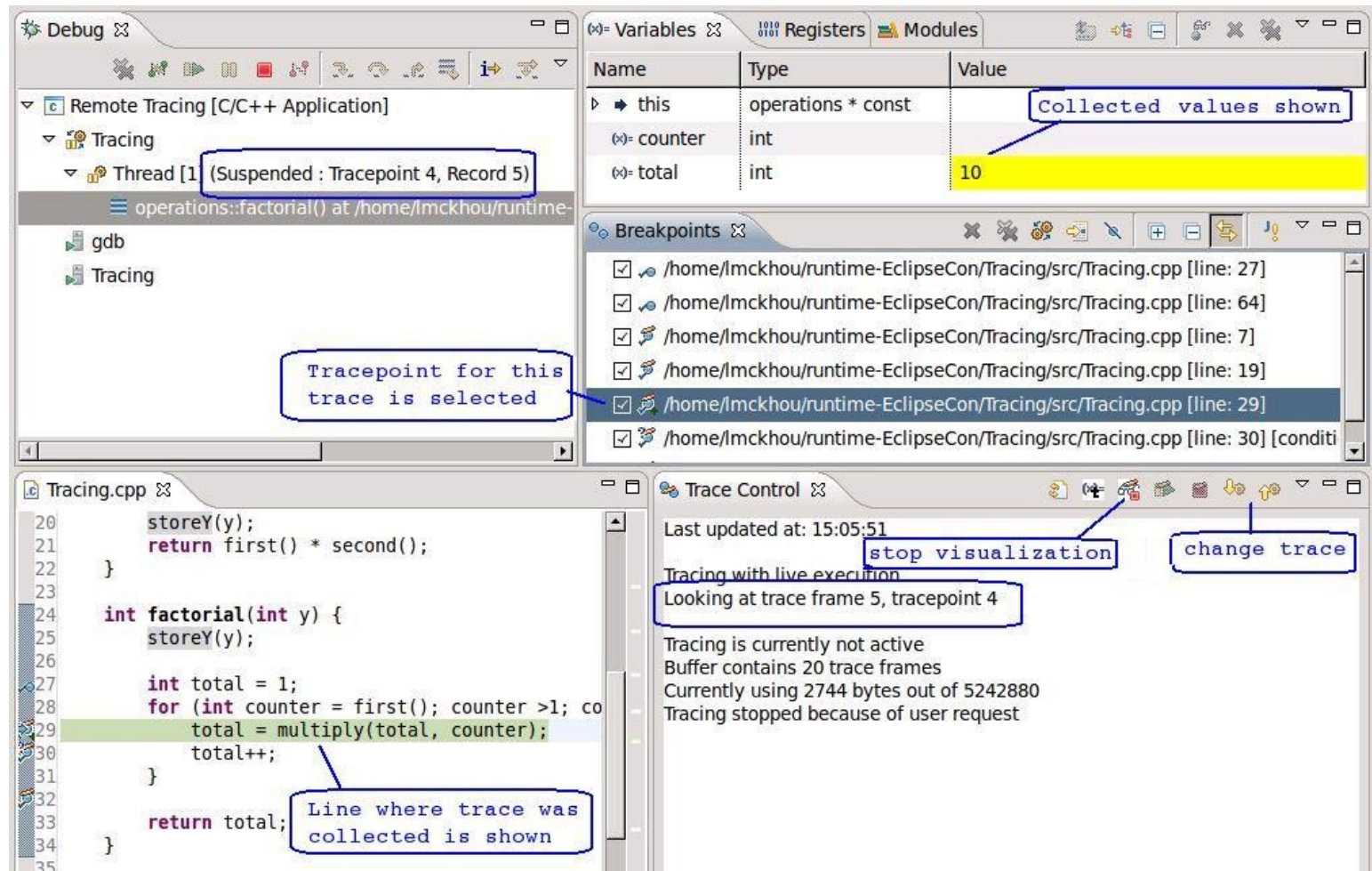
- › Resulting trace data
 - can be stored to file
 - can be visualized in Eclipse immediately or in the future



Eclipse Trace Data Visualization

- › Navigation through data records using GDB
- › Each data record is a snapshot of debug information
- › Records can be examined using standard debugger views
 - As if debugger was attached at a specific point in time
 - Only collected information can be shown
 - Highlighting of the tracepoint of interest
- › All collected data of a record can also be dumped as plain text
- › Trace data can be saved to file
- › Saved trace data can be examined offline

Eclipse Trace Data Visualization



The screenshot displays the Eclipse IDE interface during a debug session, showing trace data visualization. The interface is divided into several panels:

- Debug Console:** Shows the current thread state: "Thread [1] (Suspended : Tracepoint 4, Record 5)".
- Variables Panel:** Displays the current state of variables:

Name	Type	Value
this	operations * const	Collected values shown
counter	int	
total	int	10
- Breakpoints Panel:** Lists several breakpoints set in the file `/home/lmckhou/runtime-EclipseCon/Tracing/src/Tracing.cpp` at lines 27, 64, 7, 19, 29, and 30.
- Tracing.cpp Editor:** Shows the source code of the `factorial` function. The line `total = multiply(total, counter);` is highlighted, indicating the line where the trace was collected.
- Trace Control Panel:** Provides information about the tracing process:
 - Last updated at: 15:05:51
 - Tracing with live execution
 - Looking at trace frame 5, tracepoint 4
 - Tracing is currently not active
 - Buffer contains 20 trace frames
 - Currently using 2744 bytes out of 5242880
 - Tracing stopped because of user request

Annotations in the image highlight specific features:

- "Collected values shown" points to the `total` variable value in the Variables panel.
- "Tracepoint for this trace is selected" points to the selected thread in the Debug Console.
- "stop visualization" points to the "stop visualization" button in the Trace Control panel.
- "change trace" points to the "change trace" button in the Trace Control panel.
- "Line where trace was collected is shown" points to the highlighted line in the `Tracing.cpp` editor.

Eclipse Static Tracepoints

- › Next phase of development
- › Using GDB and UST
- › Handled like Dynamic Tracepoints, except for
 - creation
 - display list

Eclipse Static Tracepoints

- › Creation of tracepoint done by designer before compilation
- › As for Dynamic tracepoints:
 - › Enable/Disable tracepoints dynamically
 - › Dynamic condition
 - › Can additionally have dynamic tracing specified (actions)
 - › Pass count
 - › Trace-state variables

Planned Tracepoint Features

- › Support for Observer mode
- › Support for Fast Tracepoints
 - Explicit or implicit support?
- › Support for Global Actions (affecting all tracepoints)

Planned Tracepoint Features

- › Disabling tracepoints during Tracing
- › Tracepoints Enhanced Visualization:
 - Currently the user must have an idea of what has been collected
 - Goal is to directly and only show what has been collected
- › Fast Tracepoints on 3-byte instruction
 - Currently fast tracepoints are 5-byte jumps insert in the code
 - New 3-byte jump to a nearby location to the 5-byte jump

Questions?



ERICSSON