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 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

- '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
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
Eclipse Tracepoints Disassembly

› Disassembly view support for Tracepoints
› Tracepoint with condition
Eclispe Tracepoints Properties

- Tracepoints properties
  - Location
  - Enablement
  - Condition
  - Pass count
Eclipse Tracepoints Actions

**Actions for this tracepoint:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>collect total</td>
<td>Collect Action</td>
<td>collect total</td>
</tr>
</tbody>
</table>

**Available actions:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>collect total</td>
<td>Collect Action</td>
<td>collect total</td>
</tr>
<tr>
<td>collect counter</td>
<td>Collect Action</td>
<td>collect counter,$reg</td>
</tr>
<tr>
<td>Untitled Evaluate</td>
<td>Evaluate Action</td>
<td>eval $count=$count+1</td>
</tr>
</tbody>
</table>

**Buttons:**
- Attach
- New...
- Edit...
- Delete
- Restore Defaults
- Apply
- Cancel
- OK
Eclipse Tracepoints Actions

- Tracepoints action types
  - Collect
  - Evaluate
  - While-Stepping
    - Collect
    - Evaluate
Eclipse Tracepoints Variables

Trace variables can be used in tracepoint conditions or actions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Initial Value</th>
<th>Current Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$trace_timestamp</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$tracePointCounter</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Trace Control

Last updated at: 14:26:03

Tracing with live execution.
Looking at trace frame 3.
Tracing is currently not active.
Buffer contains 18 trace frames.
Currently using 2732 bytes.
Tracing stopped because of low memory.
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

- Tracepoint for this trace is selected
- Collected values shown
- Line where trace was collected is shown
- Stop visualization
- Change trace

© Ericsson | LinuxCon Tracing Mini-Summit 2010
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?