TRACING THE INNARDS OF YOUR APPLICATION

Alexandre Montplaisir, Ericsson
Florian Wininger, Polytechnique
EclipseCon 2014, PolarSys Day
PRESENTERS

› Alexandre Montplaisir
  - Software Developer, Ericsson
  - Committer on the Eclipse Linux Tools project

› Florian Wininger
  - Master student at the École Polytechnique de Montréal
  - Contributor to the Eclipse Linux Tools project

› Both work on the TMF/LTTng component
PLAN

› What is TMF, brief overview
› Advantage of data-driven components
› Data-driven views
  - Examples
› Future work
› Conclusion
› Questions
TRACING AND MONITORING FRAMEWORK

› Open-source (EPL) framework to implement trace analyzers
  - Generic interfaces, classes, views
  - Part of the Eclipse Linux Tools project (http://eclipse.org/linuxtools/)

› Reference views for:
  - LTTng kernel and UST traces
  - GDB traces
  - Custom text or XML logs

› Can be used as Eclipse plugins, or as a stand-alone application (RCP)
TRACING AND MONITORING FRAMEWORK
CURRENT VIEWS – TIME GRAPH
~1800 LoC, almost none shared with the CFV/RV!
CURRENT VIEWS – XY CHARTS
CURRENT VIEWS – SEQUENCE DIAGRAMS
STATE PROVIDERS

› Traces and log typically contain events
  − “punctual”, specific timestamp but no duration

› We often want to also track states
  − Start time
  − End time (duration)
  − State value

› Specific events indicate state transitions

› State provider : part of the framework that “converts” events to states
STATE PROVIDERS

```java
131 case LttngStrings IRQ_HANDLER_ENTRY:
132     /* Fields: int32 irq, string name */
133     {
134         Integer irqId = ((Long) content.getField(LttngStrings IRQ).getValue()).intValue();
135         /* Mark this IRQ as active in the resource tree. */
136         /* The state value = the CPU on which this IRQ is sitting */
137         quark = ss.getQuarkRelativeAndAdd(getNodeIRQs(), irqId.toString());
138         value = TmfStateValue.newValueInt(event.getCPU());
139         ss.addAttribute(ts, value, quark);
140     }/* End IRQ_HANDLER_ENTRY */
141
142     /* Change the status of the running process to interrupted */
143     quark = ss.getQuarkRelativeAndAdd(currentThreadNode, Attributes STATUS);
144     value = StateValues PROCESS_STATUS_INTERRUPTED_VALUE;
145     ss.addAttribute(ts, value, quark);
146 }
147 /* End IRQ_HANDLER_ENTRY */
148
149     /* Change the status of the CPU to interrupted */
150     quark = ss.getQuarkRelativeAndAdd(currentCPUNode, Attributes STATUS);
151     value = StateValues CPU_STATUS_IRQ_VALUE;
152     ss.addAttribute(ts, value, quark);
153 }
154 break;
155
156 case LttngStrings IRQ_HANDLER_EXIT:
157     /* Fields: int32 irq, int32 ret */
158     {
159         Integer irqId = ((Long) content.getField(LttngStrings IRQ).getValue()).intValue();
160         /* Put this IRQ back to inactive in the resource tree */
161         quark = ss.getQuarkRelativeAndAdd(getNodeIRQs(), irqId.toString());
162         value = TmfStateValue.nullValue();
163         ss.addAttribute(ts, value, quark);
164     }/* End IRQ_HANDLER_EXIT */
165     /* Set the previous process back to running */
166     setProcessToRunning(ts, currentThreadNode);
167     /* Set the CPU status back to running or "idle" */
168 ```
Advantages of using data-driven state providers and views
- Much lower barrier of entry, easier to add support for new traces, analyses and views
- Less code to maintain (especially for views)
- No need to recompile to modify the output

Making it easier to trace your application!
DATA-DRIVEN STATE PROVIDERS

- Describe state changes with an XML definition
  - Provide a way to convert events in states

```xml
<xml version="1.0" encoding="UTF-8">
<tmfxml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="xmldefinition.xsd">
  <stateProvider id="org.eclipse.linuxtools.ust.example" version="1">
    <head>
      <traceType id="org.eclipse.linuxtools.lttng2.ust.tracetype" />
      <label value="Xml kernel State System" />
    </head>

    <!-- StateValues -->
    <definedValue name="STATE_CONNECTING" value="0" />

    <!-- Event handlers -->
    <eventHandler eventName="ust_myprog:connection_wait">
      <stateChange>
        <stateAttribute type="constant" value="Threads" />
        <stateAttribute type="eventField" value="id" />
        <stateValue type="int" value="$STATE_CONNECTING" />
      </stateChange>
    </eventHandler>
  </stateProvider>
</tmfxml>
```
DATA-DRIVEN VIEWS

› Views are longer to develop
  - Data-driven views provide an easy way to configure a view for your own data.

› Currently, only the Time Graph View

› Easy to extend to XY Chart, Sequences diagrams…
DATA-DRIVEN VIEWS

<?xml version="1.0" encoding="UTF-8"?>
<tmfxml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="xmldefinition.xsd">

<stateSystemView id="org.eclipse.linuxtools.tmf.analysis.xml.sstimeview">
   <head>
      <analysis id="org.eclipse.linuxtools.ust.example" />
      <label value="Example UST View" />
   </head>
</stateSystemView>
</tmfxml>

EclipseCon 2014, PolarSys Day
EXAMPLE 1 : NETWORK CONNECTIONS

› 3 events types :
  – ust_myprog:connection_wait
  – ust_myprog:connection_start
  – ust_myprog:connection_end

› 3 states :
  – Connecting
  – Established
  – Null (Closed)

› 2 XML files, 20 lines
EXAMPLE 1: NETWORK CONNECTIONS

```c
int main(int argc, char **argv)
{
    int nb_threads = 8;
    int nb_loops = 20;
    int i;

    srand(time(NULL));

    fprintf(stderr, "Tracing...\n");

    #pragma omp parallel private(i) num_threads(nb_threads)
    for (i = 0; i < nb_loops; i++) {
        int delay1 = rand() % 50000;
        int delay2 = rand() % 50000;
        int delay3 = rand() % 50000;
        int id = omp_get_thread_num() + 1;

        /* Loop starts here */
        usleep(delay1);

        //Connection attempted
        tracepoint(ust_myprog, connection_wait, id);

        usleep(delay2);

        //Connection is established
        tracepoint(ust_myprog, connection_start, id);

        usleep(delay3);

        //Connection ends
        tracepoint(ust_myprog, connection_end, id);
    }

    fprintf(stderr, "Done.\n");
    return 0;
}
```
**EXAMPLE 1 : NETWORK CONNECTIONS**

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Source</th>
<th>Type</th>
<th>File</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:43:30.205 262 275</td>
<td>0 ust_myprog:connection_start</td>
<td>channel0_0</td>
<td>id=2</td>
<td></td>
</tr>
<tr>
<td>16:43:30.209 963 833</td>
<td>0 ust_myprog:connection_wait</td>
<td>channel0_0</td>
<td>id=7</td>
<td></td>
</tr>
<tr>
<td>16:43:30.211 405 660</td>
<td>3 ust_myprog:connection_end</td>
<td>channel0_3</td>
<td>id=5</td>
<td></td>
</tr>
<tr>
<td>16:43:30.215 602 628</td>
<td>3 ust_myprog:connection_end</td>
<td>channel0_3</td>
<td>id=6</td>
<td></td>
</tr>
</tbody>
</table>
EXAMPLE 2 : GDB

› We want to show when threads are blocked or running for a GDB debug instance.

› 4 event types :
  - gdb:inf_forked → INF_RUNNING
  - gdb:inf_stop → INF_STOPPED
  - gdb:inf_cont → INF_RUNNING
  - gdb:inf_step → INF_RUNNING
EXAMPLE 2 : GDB
EXAMPLE 3 : CHROMIUM BROWSER

› Merge UST and Kernel events in a single view.

› Another operating system, another tracer (ETW)

› See Kernel events, Chromium tasks stack, messages between Chromium thread.

› XML file : 500 lines
EXAMPLE 3 : CHROMIUM BROWSER
EXAMPLE 3: CHROMIUM BROWSER
EXAMPLE 3: CHROMIUM BROWSER

Example: Detecting bad "Wait for CPU"
FUTURE WORK

› Data-driven sequence diagram
  - Message between threads can be converted in a sequence diagram
FUTURE WORK

› Data-driven XY charts
  - State attribute can contain an integer that change during the time
FUTURE WORK

› The XML format should be seen as a configuration file.

› We are studying the possibility of using modeling software to generate the XML state provider.
CONCLUSION

› Data-driven analyses have several advantages
  – Lower effort for the user to create its own analyses
  – Less code to do new views
  – No need to install the developer version, just enjoy the product with the RCP

› Making it easier to trace your application!
REFERENCES

› http://eclipse.org/linuxtools/projectPages/lttng/
› http://lttng.org/eclipse

› Mailing list: linuxtools-dev@eclipse.org

› IRC
  - #lttng on OFTC
  - #eclipse-linux on Freenode
Evaluate This Session

1. Sign-in: [www.eclipsecon.org](http://www.eclipsecon.org)

2. Select session from schedule

3. Evaluate: +1 0 -1
QUESTIONS?