

Design of flexible analysis in TMF

Florian Winger

Presented by Geneviève Bastien

Laboratoire DORSAL
Département de génie informatique

POLYTECHNIQUE
MONTREAL

AFFILIÉE À
L'UNIVERSITÉ DE MONTREAL



Outline

- Summary of Florian's work
- Results
- Status of the development in TMF
- Future steps



Objectives

How to build custom analyses for various trace types without having to write a single line of code?

Descriptive language to do so

The screenshot displays the LTTng kernel tracing interface. The left sidebar shows a project tree with 'burnP6-16x' selected. The main window is divided into several panels:

- Resources:** A Gantt chart showing CPU usage for CPUs 0-7 and various IRQs (21, 19, 15, SOFT IRQ 3) over time. A 'Show Legend' tooltip is visible.
- Event Log:** A table of kernel events with columns for Timestamp, Channel, Event Type, and Context.
- Control Flow:** A process Gantt chart showing the execution of 'burnP6-16x-1sec' and its sub-processes.
- State System Explorer:** A table showing the state of the system at different timestamps.

Timestamp	Channel	Event Type	Context
<srch>	<srch>	<srch>	<srch>
09:47:43.052 823 204	k_2	sched_stat_wait	com
09:47:43.052 824 322	k_7	sched_switch	prev
09:47:43.052 825 718	k_3	sched_stat_wait	com
09:47:43.052 827 953	k_2	sched_switch	prev
09:47:43.052 830 468	k_3	sched_switch	prev
09:47:43.055 666 582	k_6	hrtimer_cancel	hrtim
09:47:43.055 677 757	k_6	hrtimer_expire_entry	hrtim
09:47:43.055 680 551	k_5	hrtimer_cancel	hrtim

Process	TID	PTID	09:47:42.500	09:47:43.000	09:47:43.500
ltnng-simple	9461				
ltnng	9557	9461			
burnP6-16x-1sec	9539	9461			
burnP6	9550	9539			
sleep	9556	9539			
burnP6	9555	9539			
burnP6	9552	9539			
burnP6	9551	9539			
burnP6	9554	9539			

State System / Attribute	Quark	Value at timestamp	Type	Start time	End time	Full
org.eclipse.linuxtools.lttng2.kernel.analysis						
Threads	3			09:47:42.368 794 927	09:47:43.715 051 225	Thr
9382	88			09:47:42.368 794 927	09:47:43.715 051 225	Thr
9550	241			09:47:42.368 794 927	09:47:43.715 051 225	Thr
Status	244	5	Int	09:47:43.052 824 322	09:47:43.068 067 319	Thr
System_call	245		Strin	09:47:42.712 888 101	09:47:43.715 051 225	Thr
Exec_name	243	burnP6	Strin	09:47:42.743 857 679	09:47:43.715 051 225	Thr



Goals

- **Expressiveness:** Replace actual use cases and extend to new ones
- **Usability:** Make it easy for users to create new analyses and views
- **Performance:** Preserve or improve the actual TMF performances.



- **Expressiveness:** Replace actual use cases and extend to new ones
- **Usability:** Make it easy for users to create new analyses and views
- **Performance:** Preserve or improve the actual TMF performances.

==> Choice of XML for the syntax: extensible, widely-used, easily integrates with Eclipse and TMF



XML Syntax

- Define state changes caused by events to generate a state system
- Define how to visualize it in a time graph view
- Further processing of events or state system for specific use cases through filters.

```
<stateProvider version="0" id="my.test.state.provider">
  <head>
    <traceType id="org.eclipse.linuxtools.tmf.core.development" />
    <label value="My test state provider" />
  </head>

  <definedValue name="RUNNING" value="100" />

  <eventHandler eventName="start">
    <stateChange>
      <stateAttribute type="constant" value="Tasks" />
      <stateAttribute type="eventField" value="number" />
      <stateValue type="int" value="$RUNNING" />
    </stateChange>
  </eventHandler>
</stateProvider>
```

```
<timeGraphView id="org.eclipse.linuxtools.tmf.analysis.xml.ui.views.statesystem">
  <head>
    <analysis id="my.test.state.provider" />
    <label value="My Sample XML View" />
  </head>
  <!-- StateValues -->
  <definedValue name="The process is running" value="100" color="#118811" />

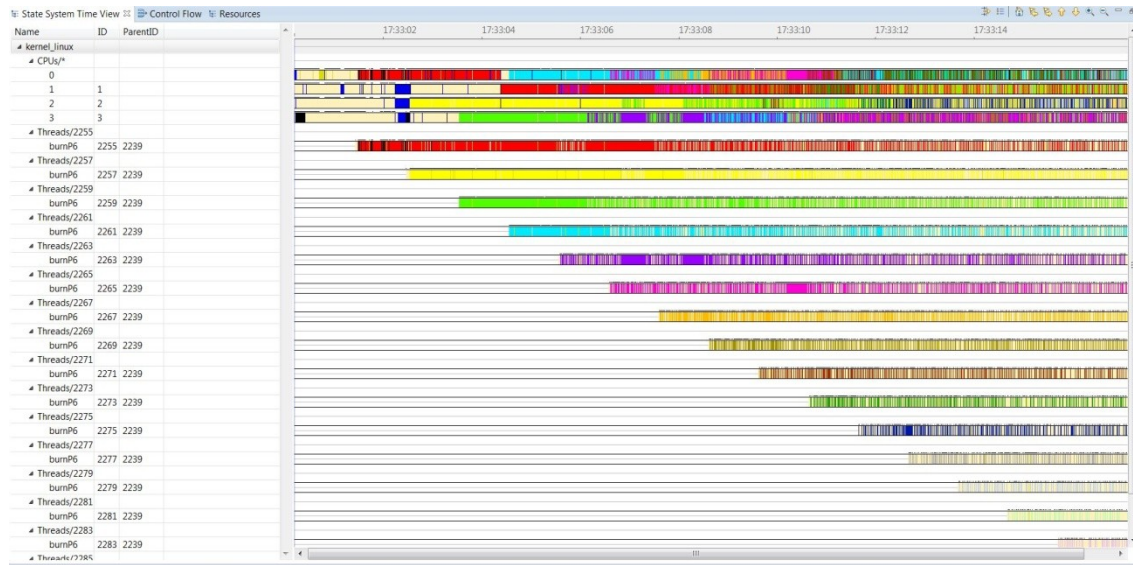
  <!-- Control Flow View -->
  <entry path="Tasks/*">
    <display type="self" />
  </entry>
</timeGraphView>
```

```
<filter name="filter_1">
  <if>
    <attribute location="App_Thread" />
    <attribute constant="Status" />
    <value int="$STATUS_WAIT_FOR_CPU" />
  </if>
  <then>
    <attribute location="Filter" />
    <attribute constant="Blocked" />
    <value int="$BLOCKED">
  </then>
  <else>
    <attribute location="Filter" />
    <attribute constant="Blocked" />
    <value int="$UNBLOCKED" />
  </else>
</filter>
```



Results (expressiveness)

- Use a single model to compare 2 different operating systems

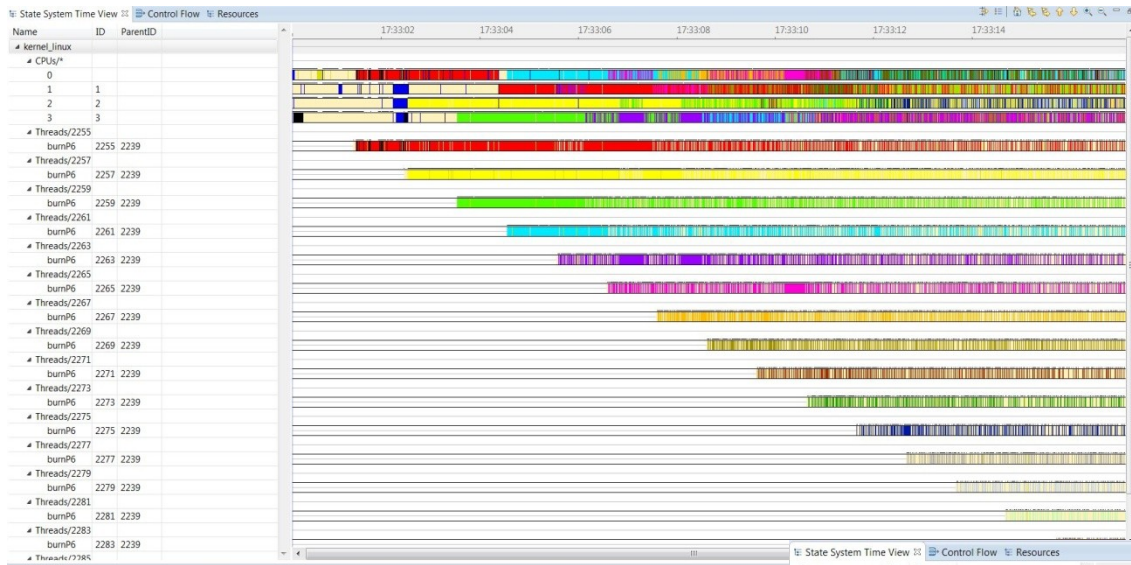


Linux: trace obtained with LTTng, same as the Lttng Kernel analysis in TMF



Results (expressiveness)

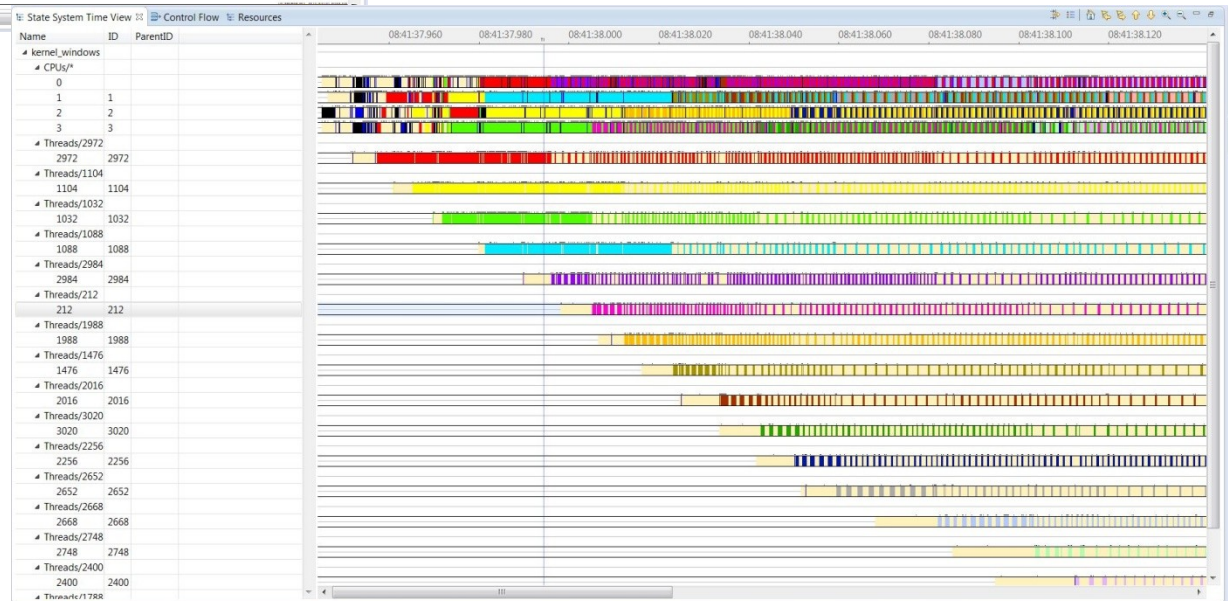
- Use a single model to compare 2 different operating systems



Linux: trace obtained with LTTng, same as the Lttng Kernel analysis in TMF

Windows: trace obtained with ETW and converted to CTF using ETW2CTF converter

<https://github.com/fwininger/ETW2CTF/>



Results (usability)

State providers	Java	XML
Development	State provider class + analysis class + plugin.xml	XML state provider element
Development environment	Eclipse SDK + TMF development environment	XML (text) editor, one will come with TMF.
Testing and debugging	Compile + execute (2 nd Eclipse instance) + state system explorer	Import modified XML file + state system explorer

Views	Java	XML
Development	View class + presentation provider class + entry class + plugin.xml	XML view element (~10 lines for a simple view)
Flexibility	Fully customizable	Within limits of implemented features



Results (performance)

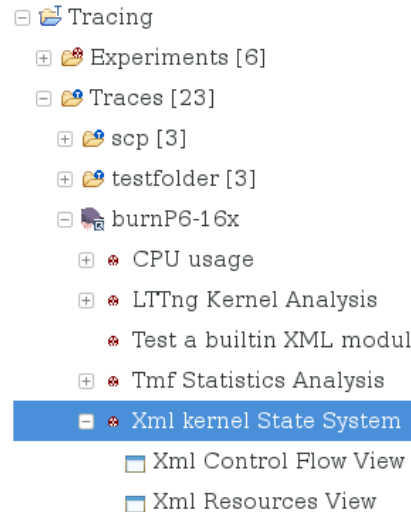
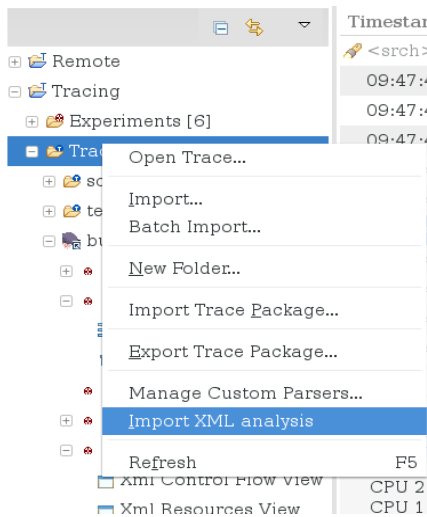
- Build time of the model

Trace 100 MB	Java	XML
Average time (s)	49.359	50.025
Standard deviation (s)	1.034	1.140
Min _(s)	47.054	44.325
Max _(s)	52.670	52.427

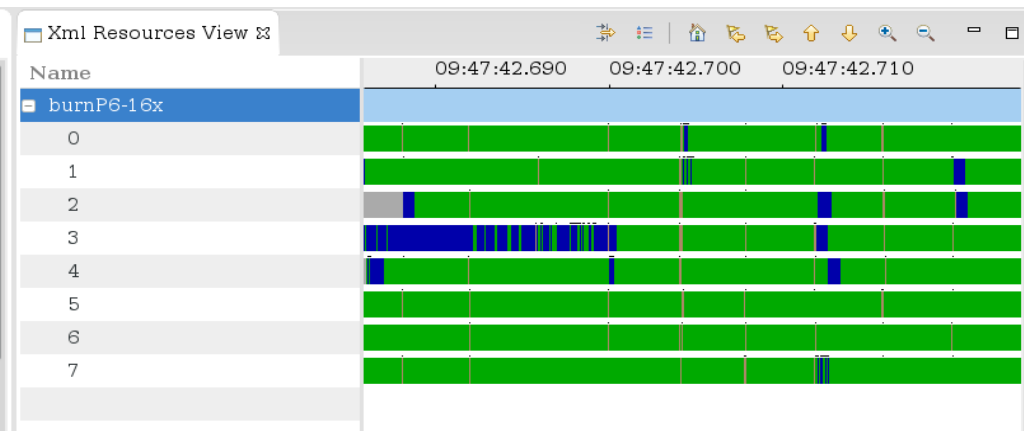
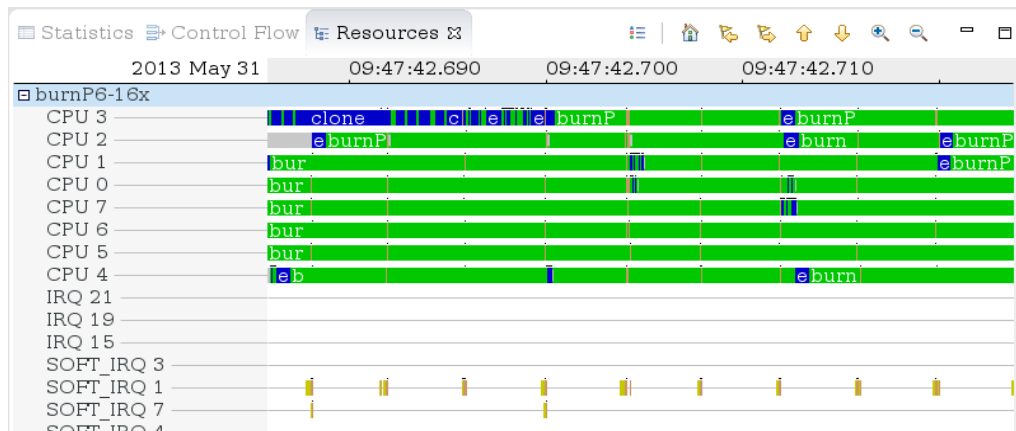


Status in TMF

- State provider and Time graph views are in TMF master

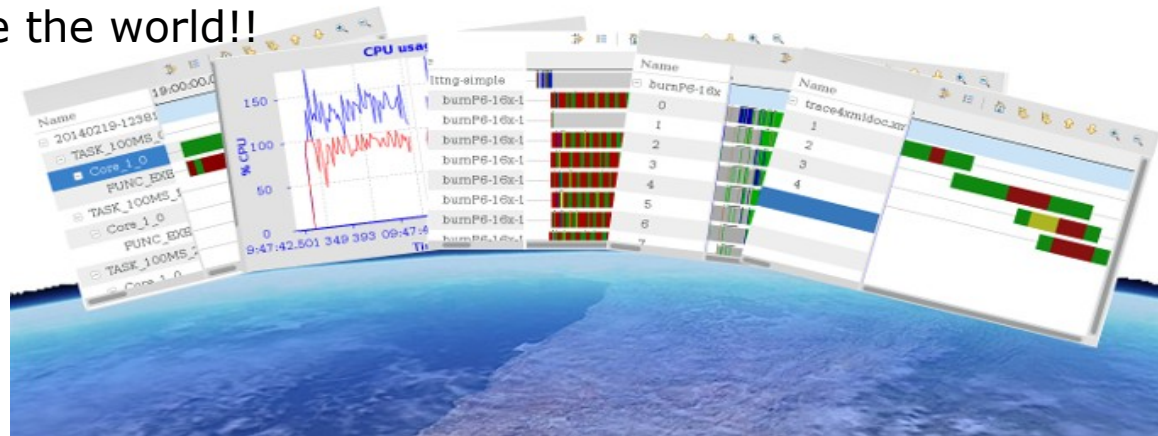


- ▢ Data driven analysis
 - ▢ Importing an XML file containing analysis
 - ▢ Defining XML components
- ▢ Defining an XML state provider
 - ▢ Definitions and example
 - ▢ Determining the state system structure
 - ▢ Writing the XML state provider
 - ▢ Debugging the XML state provider
- ▢ Defining an XML time graph view



Future steps

- Short term:
 - Views: support tooltips and texts in states, XY charts
 - State providers: Add expressiveness (operations on string, regexes, save values for later use, etc.)
 - Make sure it works for experiments with different trace types
- Medium term:
 - Implement filters described by Florian
 - Generate XML state providers from UML state diagrams
- Long term:
 - Multiply analyses and dominate the world!!



Questions

