ÉCOLE POLYTECHNIQUE MONTRÉAL

Design of flexible analysis in TMF
XML State-Provider and XML Filter

Presented by : Florian Wininger

DORSAL - December 10th 2013

Introduction

- Objectives :
  - Capture and merge observations made anywhere in the system (alerts, events, and states) by many surveillance systems (AV, HIDS, performance monitor systems, software tracers…)
  - Convert this data in the most appropriate format for detection analysis.

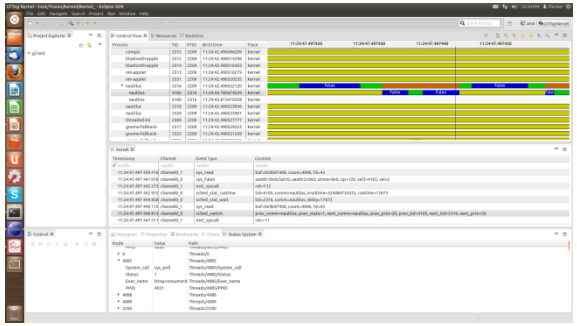
- Proposed solution :
  - Formal expressions to use the state-system in TMF
    - Allow the creation of personalized State System
    - Allow a new analysis with virtual states (filter)

Motivation

- Improve the use of the State System

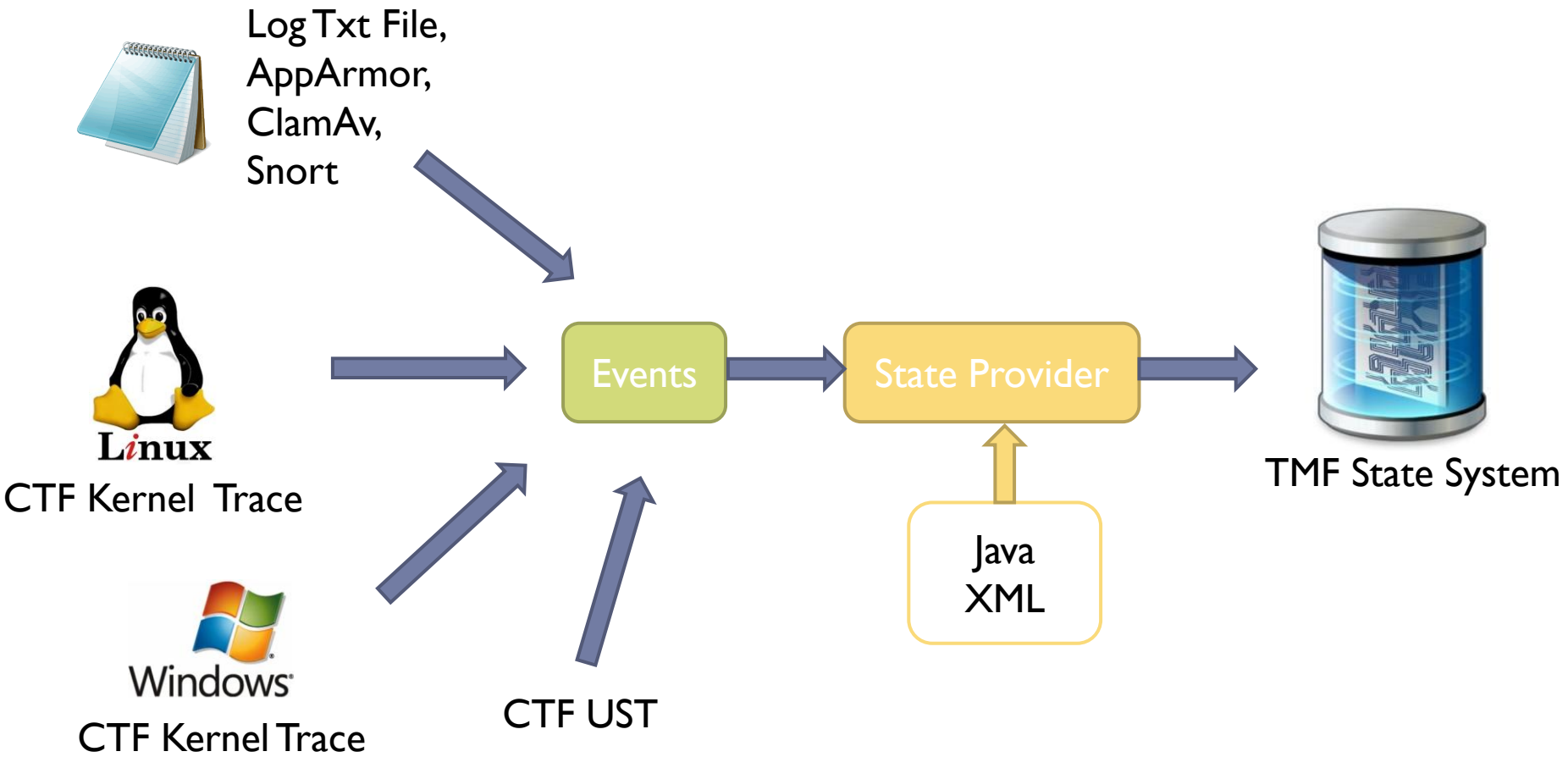
CTF (Linux)
Kernel Trace

State System



TMF Control Flow View

Motivation


Log Txt File,
AppArmor,
ClamAv,
Snort
Linux
CTF Kernel Trace
Windows
CTF Kernel Trace
CTF UST
Events
State Provider
Java
XML
TMF State System

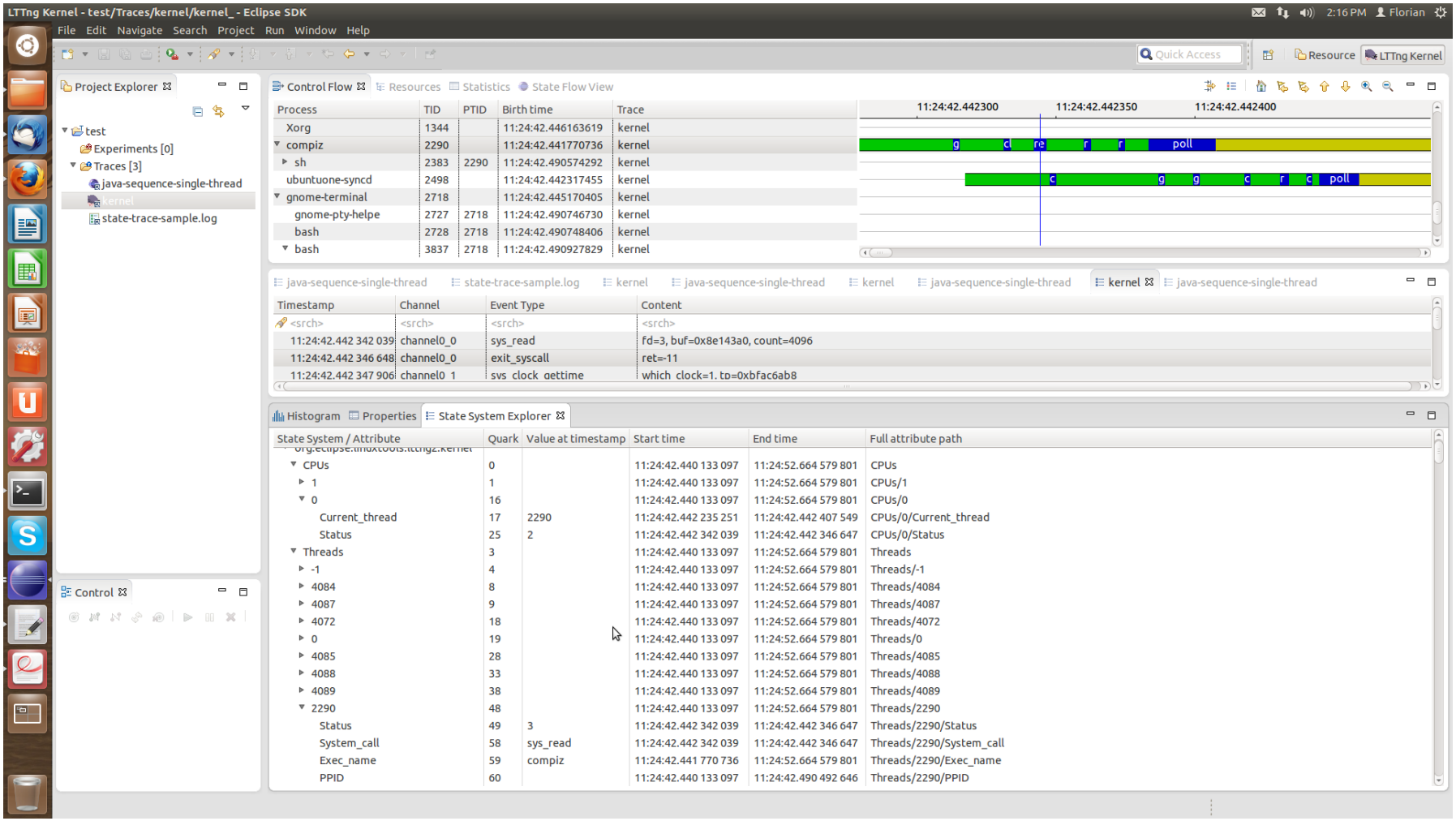
Motivation

- Currently the way to do the state system is hardcoded in a java file : CtfKernelStateInput.java with a big switch/case

- The idea is to transcribe it into a XML style sheet and after that we can have an interface to generate the XML.

- Multi-level information
  - We can merge multiple data's source (examples UST-Kernel, VM-Host…)

State System Explorer



Very interesting use case

- Windows Kernel Trace
  - ETW : Event Trace for Windows
  - .etl : etl format
  - Can be use for Kernel and UST !

- Google want to optimize Chromium
  - Convert ETW trace to CTF

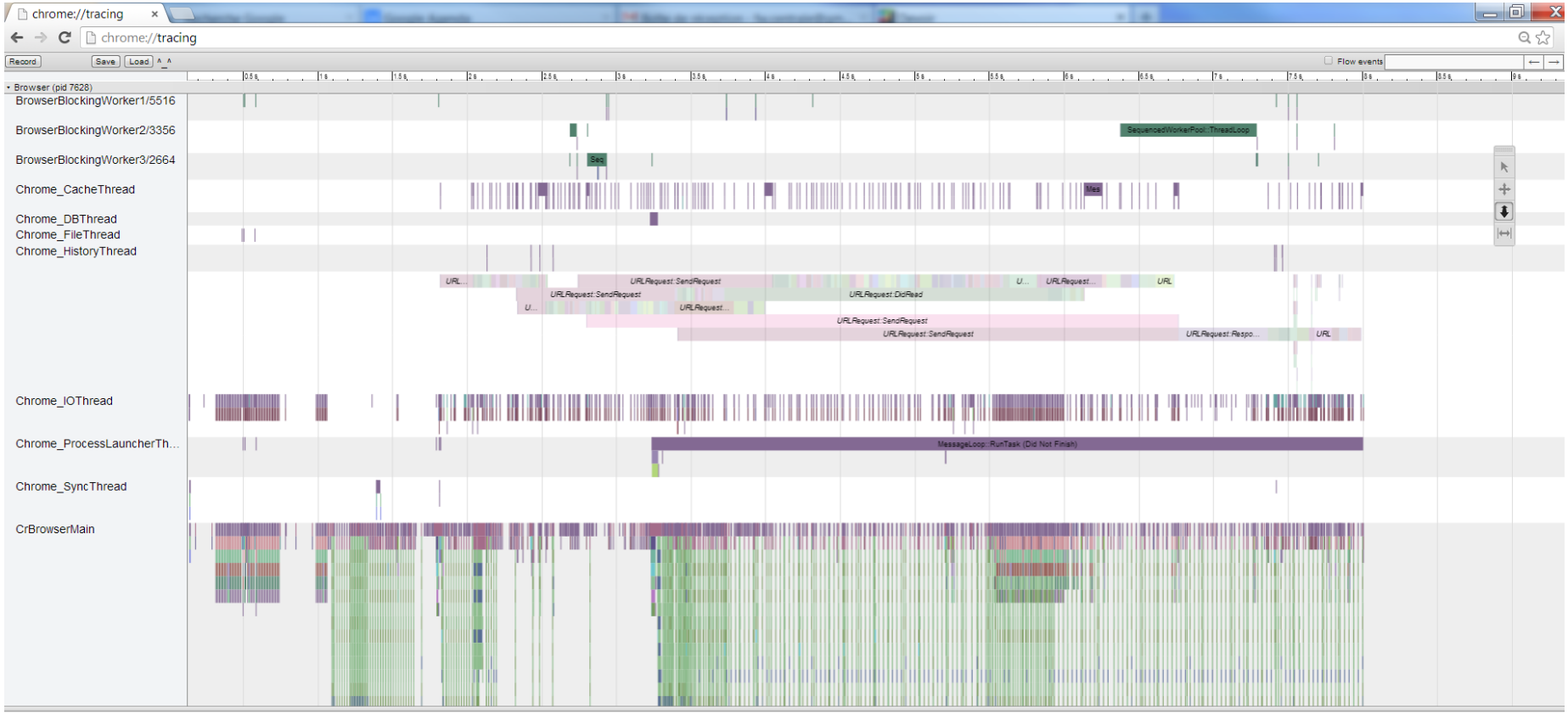
- Windows Performance Toolkit

Work – Chromium project

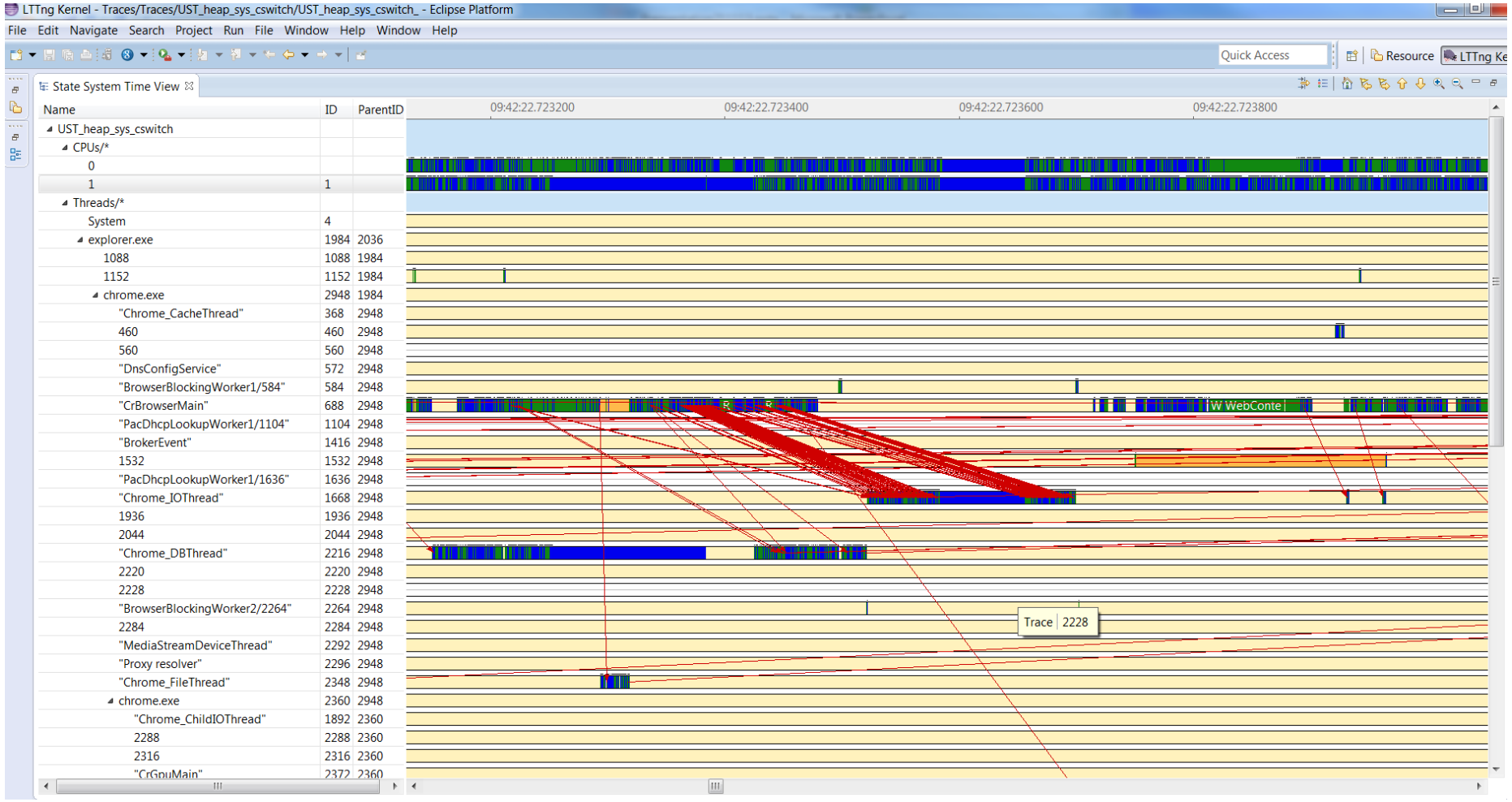
- ETW2CTF convertor
  - Allow to convert Windows Trace Format in CTF
  - https://github.com/fwininger/ETW2CTF/
- Patch in Chromium
  - Use ETW to record the existing trace points
  - Currently only for windows, but in the future, we can use LTTng-UST to extract the same information for Linux
- Thanks to Etienne Bergeron and François Doray

Work – Chromium project

- Chrome://tracing



Work – Chromium project



Work – Chromium project - Advantage

- More information :
  - Multi-level : UST and Kernel!
  - Example : System Call, Wait for CPU, …
  - Memory event (alloc and free)
    - Prototype of statistics of cumulative allocation and free in a time range.
- Better scalability
  - Chrome://tracing use JSON and Java Script, in memory solution

Work – Chromium project - Limitations

- Currently the complete toolchain to capture and view the trace is slow, but we are working hard to change that!
  - Produce 50 MB/s of events
  - Conversion ETW to CTF : 4-5 MB/s (but can be better if we use less the WPT API)
  - Build of the state system in TMF : 2-3 MB/s

Work – Chromium project - Limitations

- Scalabily problems for the UI !
  - The State System scales to big traces, but not how it is currently used by the UI.
  - Cache added [last week] : 30x acceleration
  - Add a new query type, better suited for backend performance (automatic range query) [in review]
  - Avoid long tree branches with single child: 10x acceleration for queries when there is a large number of attributes in the current state [in review]

Work – Make your own State System

- Example : UST trace (tracepoint added in GDB)
  - https://github.com/fwininger/XML4TMF
- We want to know when the GDB is running
- 2 states :

```
<stateValue name="INF_RUNNING" value="1" />
<stateValue name="INF_STOPPED" value="0" />
```

Work – Make your own State System

- 1 location :

```
<location id="GDB_Status">
        <attribute constant="gdb" />
        <attribute eventfield="pid" />
        <attribute constant="Status" />
</location>
```

- Example of state change :

```
<eventHandler eventname="gdb:inf_forked">
        <stateChange>
                <attribute location="GDB_Status" />
                <value int="$INF_RUNNING" />
        </stateChange>
</eventHandler>
```

Work – Make your own State System

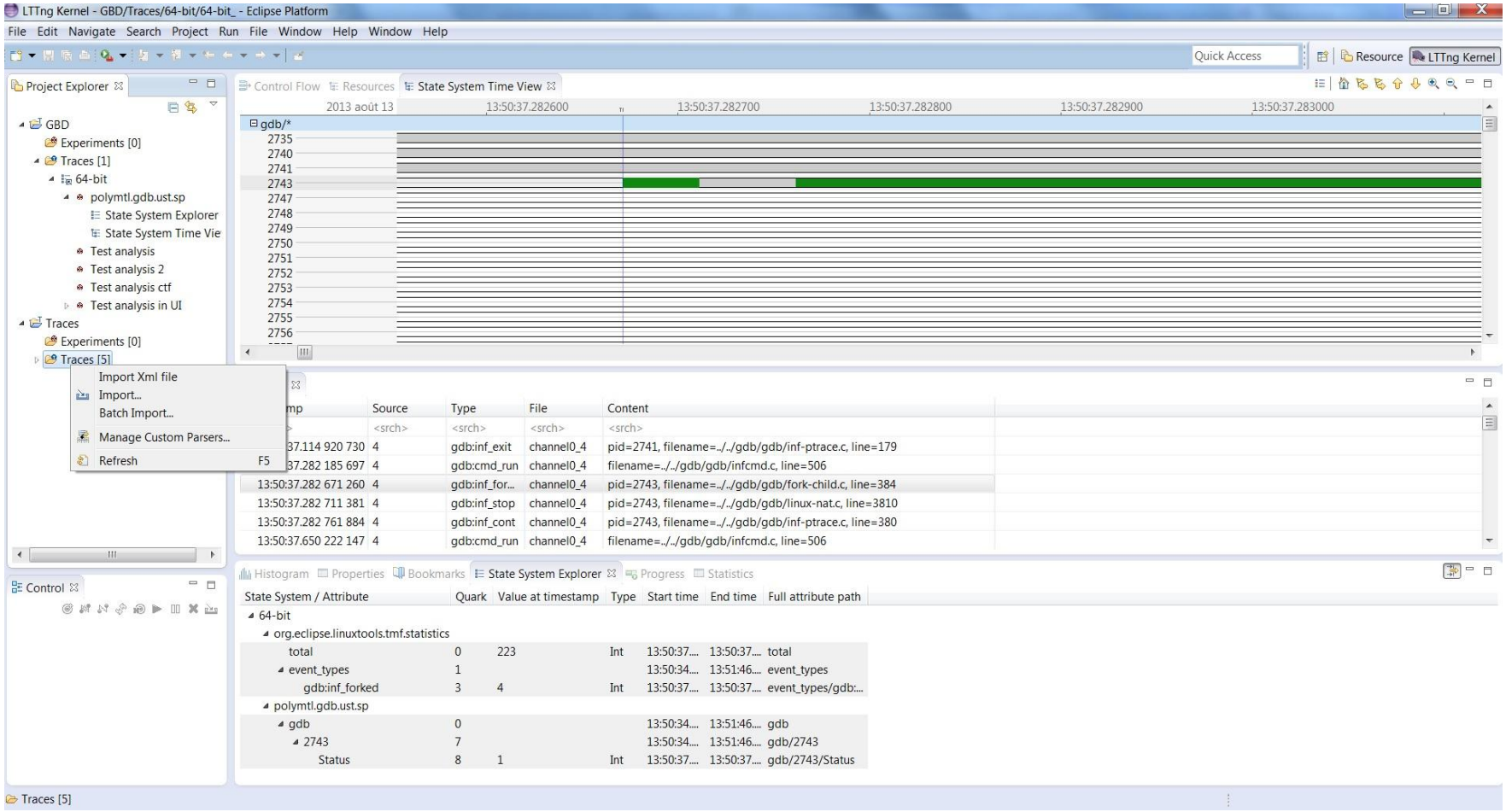
```
19      <view id="org.eclipse.linuxtools.tmf.analysis.xml.sstimeview">
20              <head>
21                      <analysis id="polymtl.gdb.ust.sp" />
22              </head>
23              <!-- StateValues -->
24              <stateValue name="INF_RUNNING" value="1" color="#118811" />
25              <stateValue name="INF_STOPPED" value="0" color="#CCCCCC" />
26
27              <!-- Control Flow View -->
28              <line id="gdb/*" display="Status" />
29
30      </view>
31      <stateprovider analysisid="polymtl.gdb.ust.sp">
32              <head>
33                      <tracetype id="org.eclipse.linuxtools.tmf.ui.type.ctf" />
34                      <view id="org.eclipse.linuxtools.tmf.analysis.xml.sstimeview" />
35              </head>
36
37              <!-- StateValues -->
38              <stateValue name="INF_RUNNING" value="1" />
39              <stateValue name="INF_STOPPED" value="0" />
40
41              <location id="GDB_Status">
42                      <attribute constant="gdb" />
43                      <attribute eventfield="pid" />
44                      <attribute constant="Status" />
45              </location>
46
47              <eventHandler eventname="gdb:inf_forked">
48                      <stateChange>
49                              <attribute location="GDB_Status" />
50                              <value int="$INF_RUNNING" />
51                      </stateChange>
52              </eventHandler>
53              <eventHandler eventname="gdb:inf_stop">
54                      <stateChange>
55                              <attribute location="GDB_Status" />
56                              <value int="$INF_STOPPED" />
57                      </stateChange>
58              </eventHandler>
59              <eventHandler eventname="gdb:inf_cont">
60                      <stateChange>
61                              <attribute location="GDB_Status" />
62                              <value int="$INF_RUNNING" />
63                      </stateChange>
64              </eventHandler>
```

View's parameters

Analysis parameters

State system definition

Work – Make your own State System



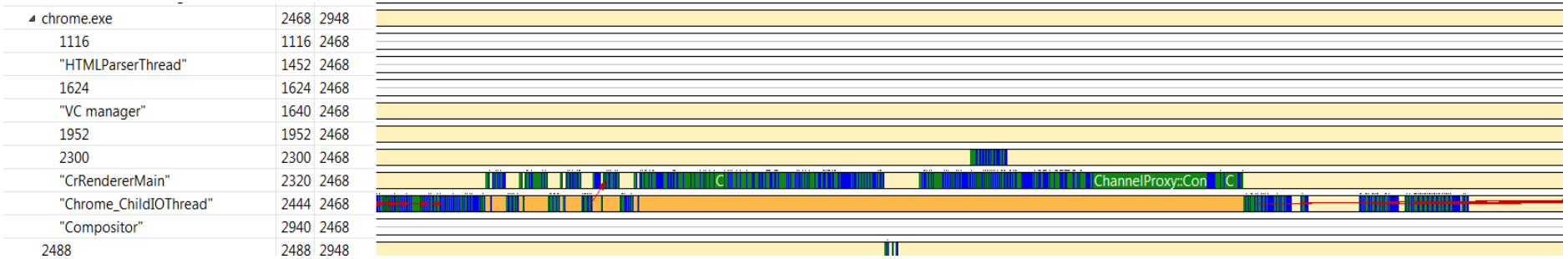
Work - Filtering

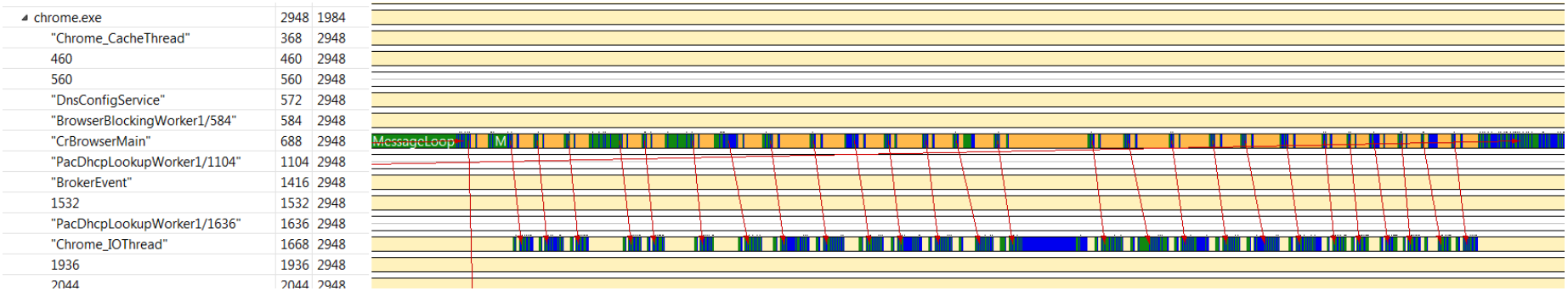
- It's very easy to extend the XML definition to make filters (based on finite state machine)


```
<filter name="wait">
    <transition start=0 end=1>
            <if>
                <condition>
                        <attribute localion="Chrome" />
                        <value int="$PROCESS_STATUS_WAIT_FOR_CPU" />
                </condition>
            </if>
    </transition>
    <transition start=1 end=0>
            <if>
                <condition>
                        <not>
                            <attribute localion="Chrome" />
                            <value int="$PROCESS_STATUS_WAIT_FOR_CPU" />
                        </not>
                </condition>
            </if>
    </transition>
</filter>
```

Work - Filtering

- Example : Detecting bad Wait for CPU





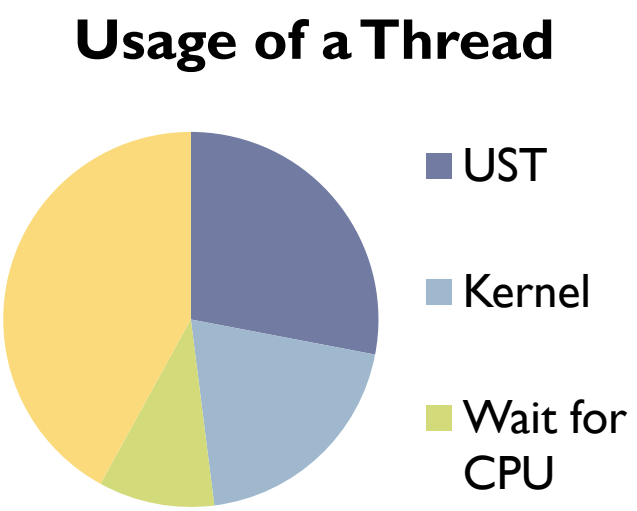
Work - Filtering

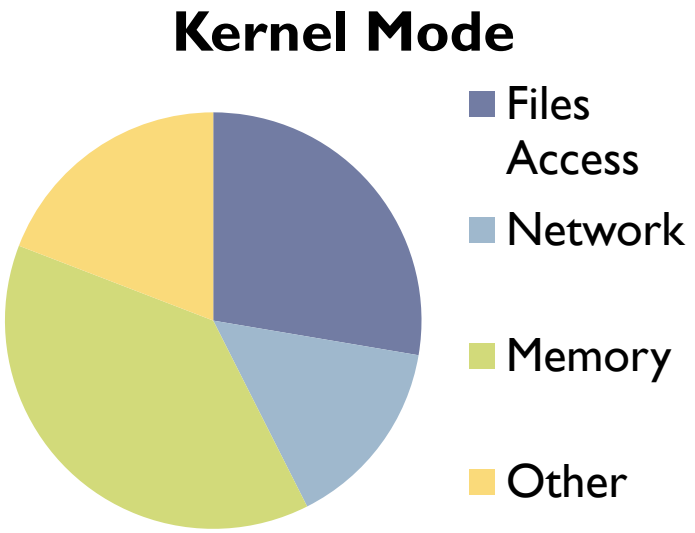
- We add a level of abstraction
- We can do the same filter based on event but it's more complex.
  - Cswitch event with the event field OldThreadWaitReason=32 means that the thread is preempted.
- Easy way to add a trigger to detect problems.

Work – Filtering

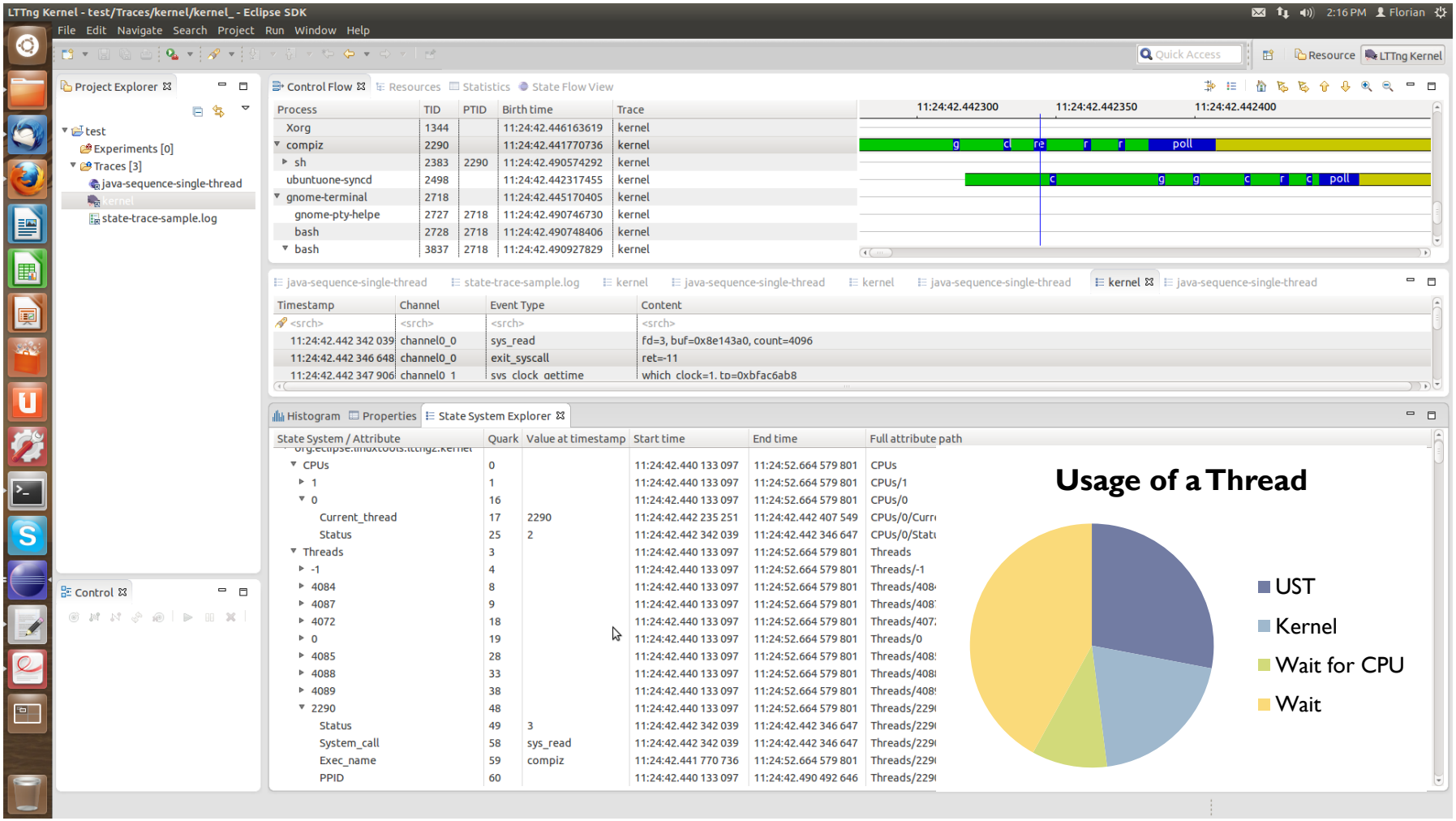
- In fact we add virtual state interval with the filter.

- We can use these filters to add statistics. (based on time or frequency…)





Work - Filtering



Conclusion

- Formal expressions to have a more efficient use of the state-system in TMF
- Good way to easily use external data and make specific and flexible analysis
- More users, more contributors

Questions