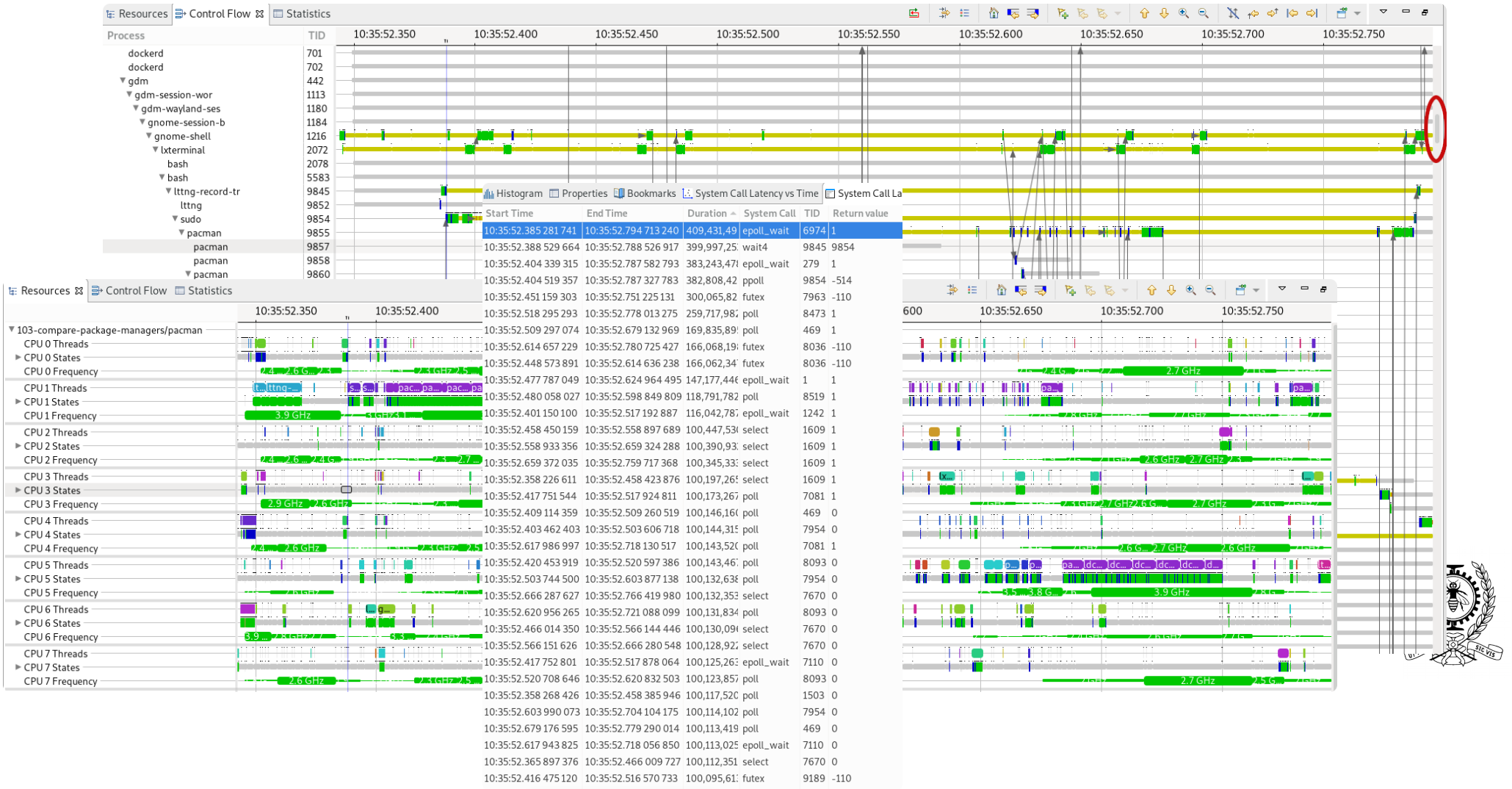
Trace Compass Filtering and Scripting with EASE

Progress Report Meeting,
École Polytechnique de Montréal
May 6, 2019

Geneviève Bastien
Research Associate

New feature #1: Global Filtering

Problem? Too much information!



New feature #1: Global Filtering

* Apply filters to all views

* Filters use a Language Server Protocol

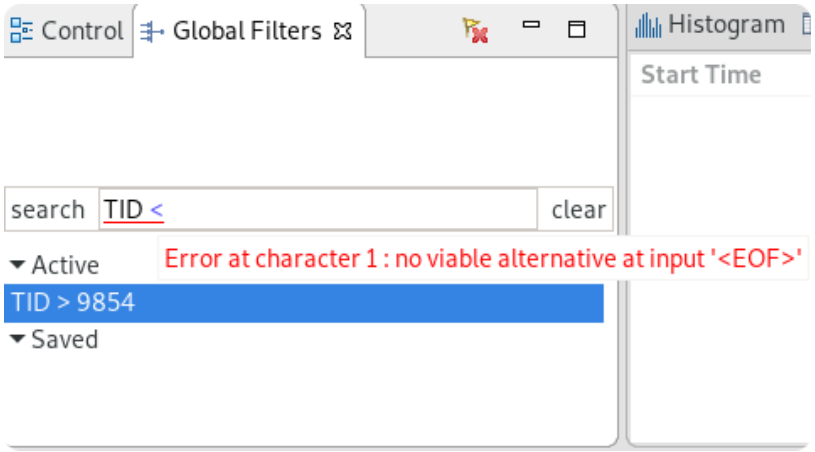
    -> Validity check as you type

    -> Syntax highlighting

    -> Eventual autocompletion, “smart” filters

    -> Can be used as is by theia

* Available from the Incubator: “Global filters” feature



New feature #1: Global Filtering

* Views on which filters apply:

  -> Time Graphs

  -> Segment tables / scatter chart

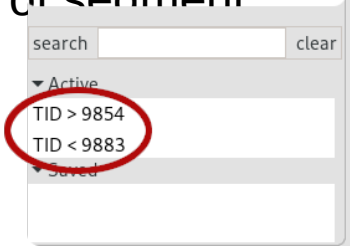
  -> Events editor

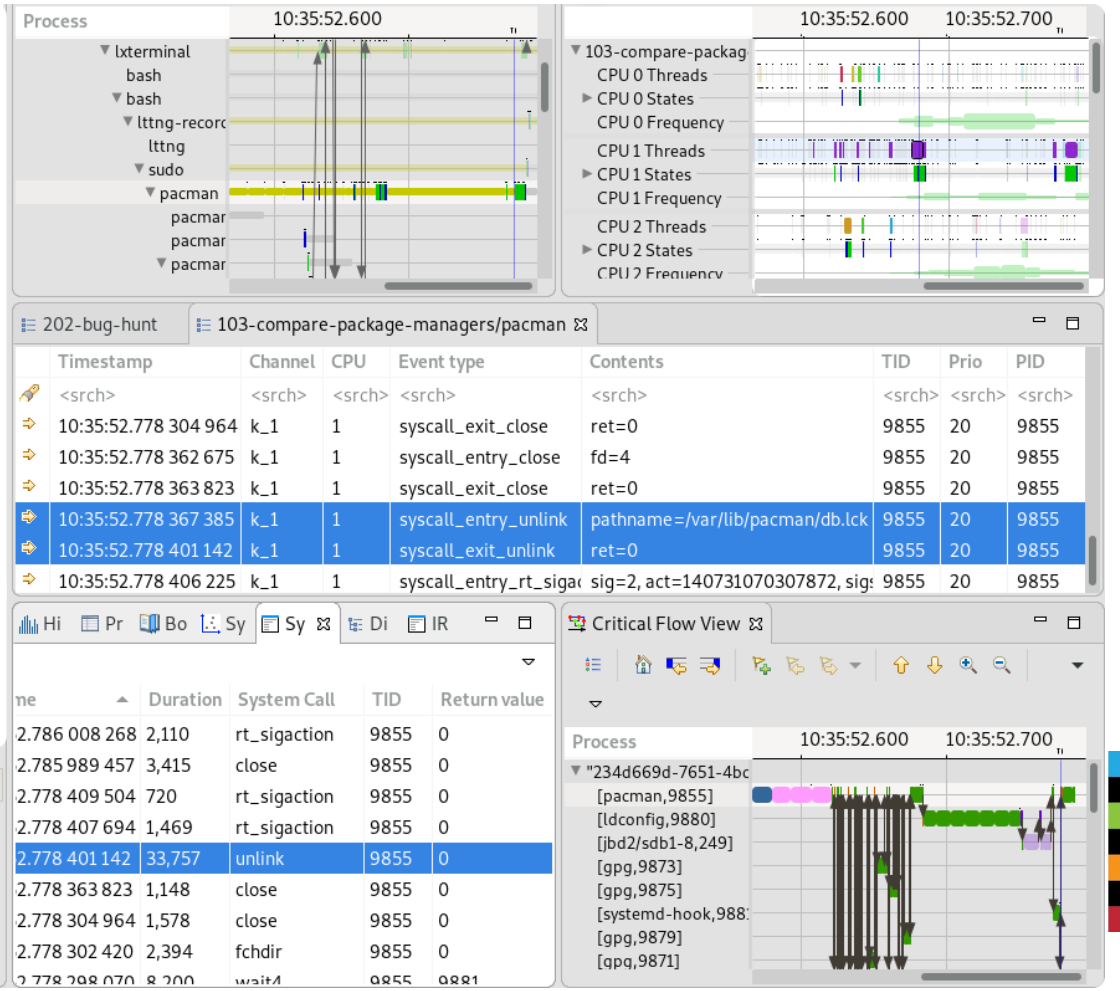
* Tested and known to work on:

  -> Common fields: TID, PID, CPU

  -> Visible tooltip fields in time graphs

  -> Additional columns of segment tables





New feature #1: Global Filtering

* Future work :

  -> Add smarter filters with namespace: “thread.name matches java”

  -> Specify what to do with individual filters: highlight, hide what does not match

  -> Proper UX: fine tune filter application per view

HELP US!! We need user feedback

New feature #2: Scripted Analyses

Problem?

* Finite number of available analyses

* Option: XML

  -> Very verbose
  -> Hard to read
  -> Hard to debug

  -> But it works!

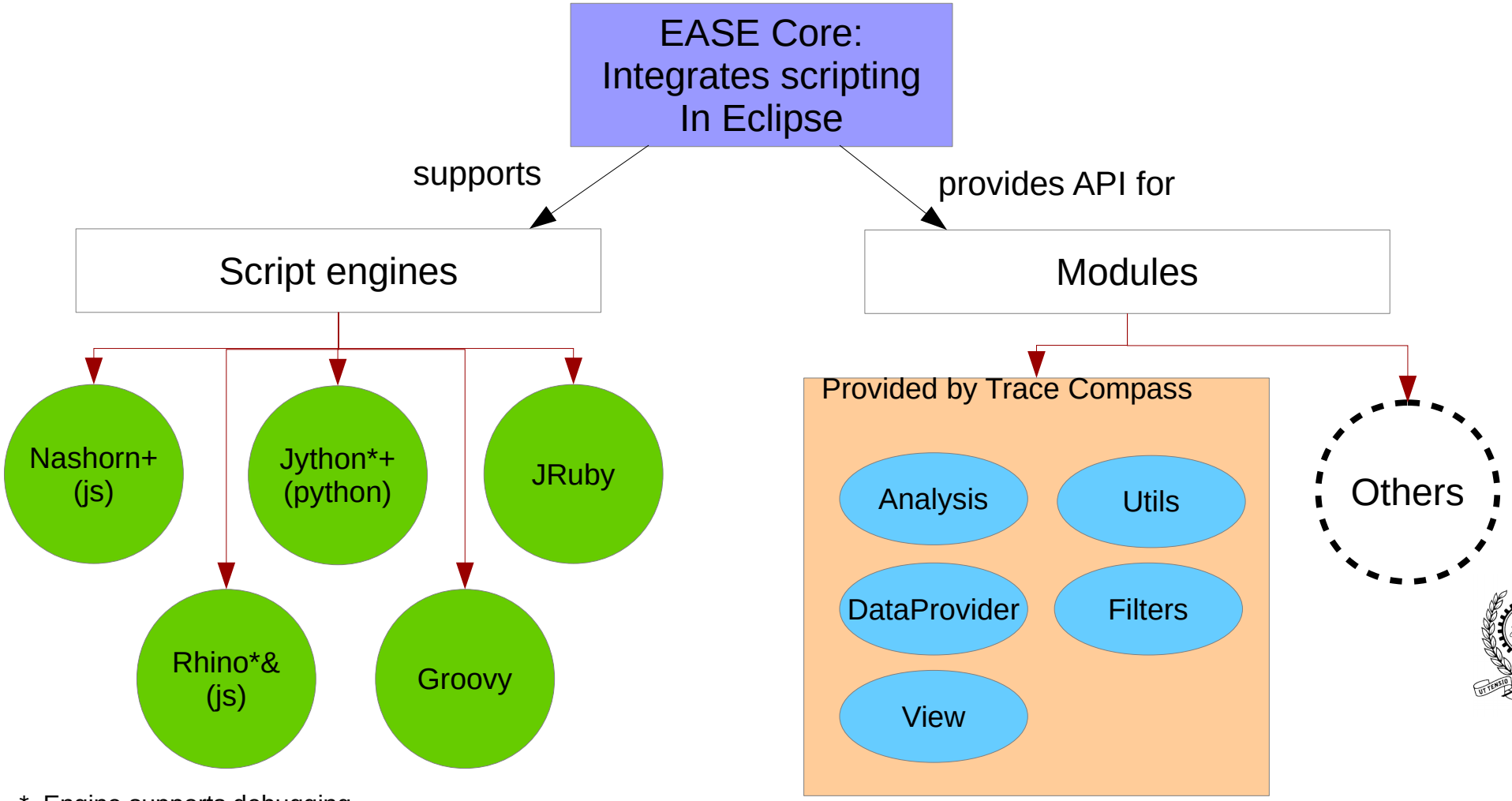
* Developers want to do what they know:

  Develop!

```
        <stateChange>
            <stateAttribute type="constant" value="#CurrentScenario" />
            <stateAttribute type="constant" value="type" />
            <stateValue type="eventField" value="evName" />
        </stateChange>
        <stateChange>
            <stateAttribute type="constant" value="#CurrentScenario" />
            <stateAttribute type="constant" value="component" />
            <stateValue type="script" value="cat == null ? 'UNKNOWN' : cat" scriptEngine="nashorn" >
                <stateValue id="cat" type="eventField" value="cat" />
            </stateValue>
        </stateChange>
    </action>

    <action id="push_event_type">
        <!-- Push the current event to the thread's callstack -->
        <stateChange>
                <stateAttribute type="location" value="CurrentThread" />
                <stateAttribute type="constant" value="CallStack" />
                <stateValue type="eventField" value="evName" stack="push"/>
        </stateChange>
        <stateChange>
                <stateAttribute type="location" value="CurrentThread" />
                <stateAttribute type="constant" value="cpu" />
                <stateValue type="eventField" value="cpu" />
        </stateChange>
    </action>

    <action id="pop_event_type">
        <!-- Pop the current event from the callstack -->
        <stateChange>
                <stateAttribute type="location" value="CurrentThread" />
                <stateAttribute type="constant" value="CallStack" />
                <stateValue type="eventField" value="evName" stack="pop"/>
        </stateChange>
    </action>

    <!-- FSMs -->

    <fsm id="tgThread" initial="Wait_thread_start">
        <state id="Wait_thread_start">
            <!-- The state will stay here until we have a thread start event -->
            <transition event="*" cond="is_start" target="in_thread" action="entering_thread:push_event_type"/>
        </state>
        <state id="in_thread" >
            <!-- The state will loop on itself until the thread ends and increment the operations that happen during the execution -->
            <transition event="*" cond="thread_thread:is_start" target="in_thread" action="push_event_type"/>
            <transition event="*" cond="thread_thread:is_end:last_out" target="end_thread" action="pop_event_type"/>
            <transition event="*" cond="thread_thread:is_end" target="in_thread" action="pop_event_type"/>
        </state>
```

New feature #2: Scripted Analyses

Eclipse Advanced Scripting Environment (EASE)



* Engine supports debugging
\+ Tested and working
& Some module functions have problems

New feature #2: Scripted Analyses

* First EASE patch:
  February 2015!!! (abandoned)

* Prototype done: run analyses + show views, even with arrows!

* Patches on gerrit, pending CQs

* Temporary update site available until patches are accepted

* Theoretically, one can do ANYTHING in java if they know how

* In practice, we need to provide proper APIs

* APIs can be added with relative EASE

```
// Create an analysis named activetid.js.
var analysis = getAnalysis("activetid.js");

if (analysis == null) {
    print("Trace is null");
    exit();
}

// Get the analysis's state system so we can fill it, false indicates to create a new state system even if one already exists, true wo
var ss = analysis.getStateSystem(false);

// The analysis itself is in this function
function runAnalysis() {
    // Get the event iterator for the trace
    var iter = analysis.getEventIterator();

    var event = null;
    // Parse all events
    while (iter.hasNext()) {

        event = iter.next();

        // Do something when the event is a sched_switch
        if (event.getName() == "sched_switch") {
            // This function is a wrapper to get the value of field CPU in the event, or return null if the field is not present
            cpu = getFieldValue(event, "CPU");
            tid = getFieldValue(event, "next_tid");
            if ((cpu != null) && (tid != null)) {
                // Write the tid to the state system, for the attribute corresponding to the cpu
                quark = ss.getQuarkAbsoluteAndAdd(cpu);
                // modify the value, tid is a long, so "" + tid make sure it's a string for display purposes
                ss.modifyAttribute(event.getTimestamp().toNanos(), "" + tid, quark);
            }
        }

    }
    // Done parsing the events, close the state system at the time of the last event, it needs to be done manually otherwise the state
    if (event != null) {
        ss.closeHistory(event.getTimestamp().toNanos());
    }
}

// This condition verifies if the state system is completed. For instance, if it had been built in a previous run of the script, it wo
if (!ss.waitUntilBuilt(0)) {
    // State system not built, run the analysis
    runAnalysis();
}

// Get a time graph provider from this analysis, displaying all attributes (which are the cpus here)
provider = createTimeGraphProvider(analysis, {'path' : '*'});
if (provider != null) {
    // Open a time graph view displaying this provider
    openTimeGraphView(provider);
}
```

Demo

-> https://secretaire.dorsal.polymtl.ca/~gbastien/screenshots/filters.mkv

-> https://secretaire.dorsal.polymtl.ca/~gbastien/screenshots/scripting.mkv

Questions ?

Resources

- Temporary update site for scripting features (until CQ done): https://secretaire.dorsal.polymtl.ca/~gbastien/TracingRCP/IncubatorUpdateSite/
- Information on EASE: https://www.eclipse.org/ease
- Example scripts: https://secretaire.dorsal.polymtl.ca/~gbastien/traces/scripts/
- My personal blog on new features: http://versatic.net
- Twitter: @genbastien