I will now talk about my experiences in User tracing for over 20 minutes

Matthew Khouzam - Ericsson

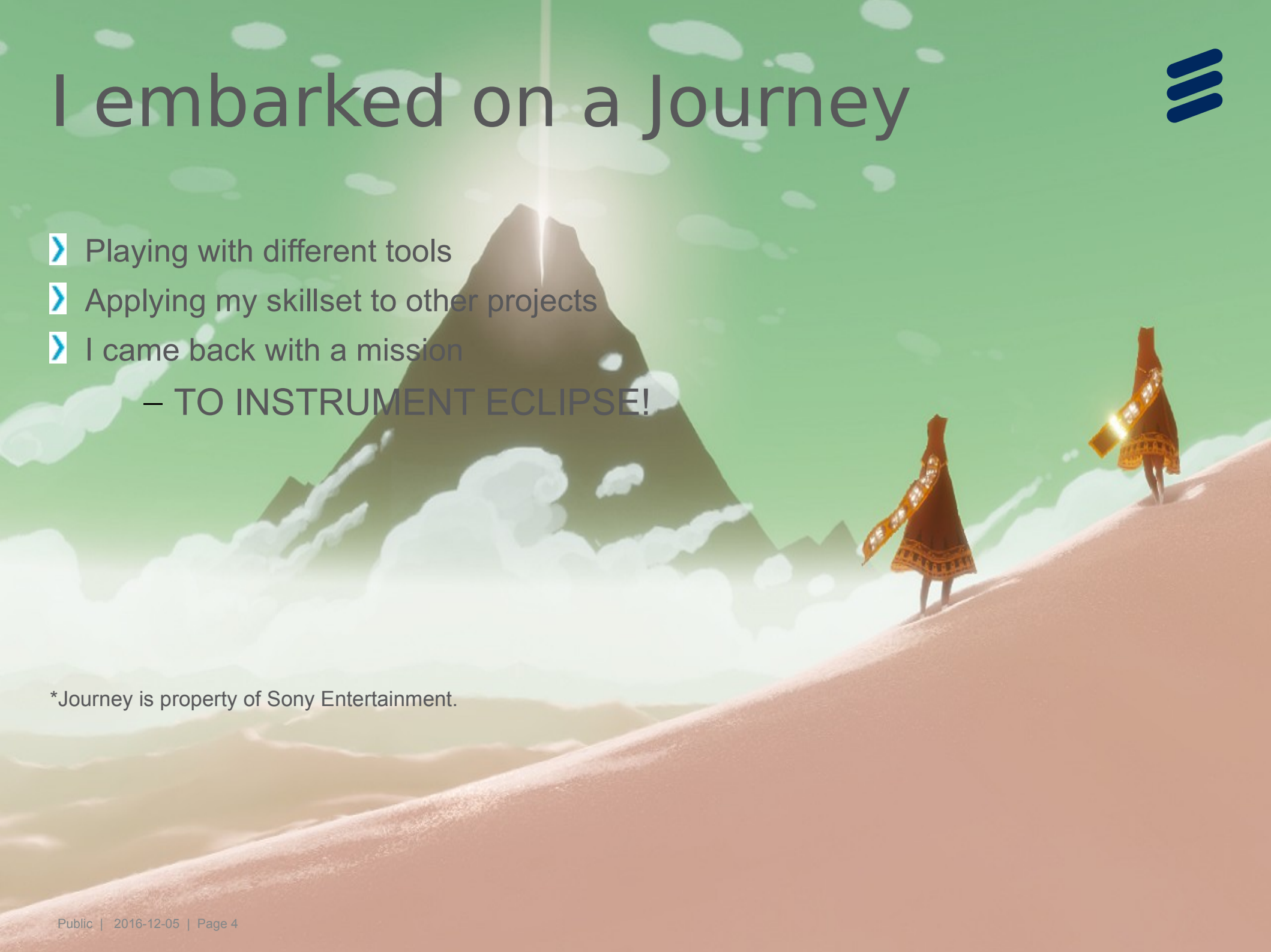
Technologies used

- Call Stack Analysis
- Call Graph Analysis
- Descriptive Statistics
- Trace Parser (custom)
- Call Stack View
- Flame Graph View
- XML Analyses

Story Time

- I love creating, I am high energy, I am self motivated
- Sometimes this can be an opportunity
- I currently have 200 pending patches (gerrit / github)
- I was suggested to work on another project

I embarked on a Journey

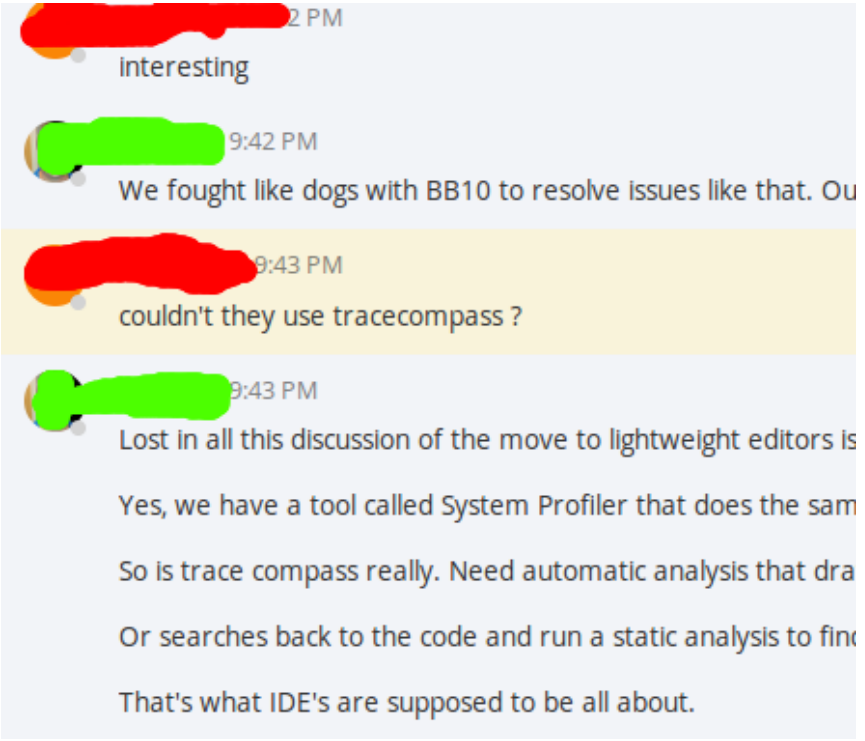


- Playing with different tools
- Applying my skillset to other projects
- I came back with a mission
  - TO INSTRUMENT ECLIPSE!

*Journey is property of Sony Entertainment.

My finding

- There is no good complete solution to the tracing problem yet.
- Eclipse developers want to use Trace Compass, but can’t at this moment.

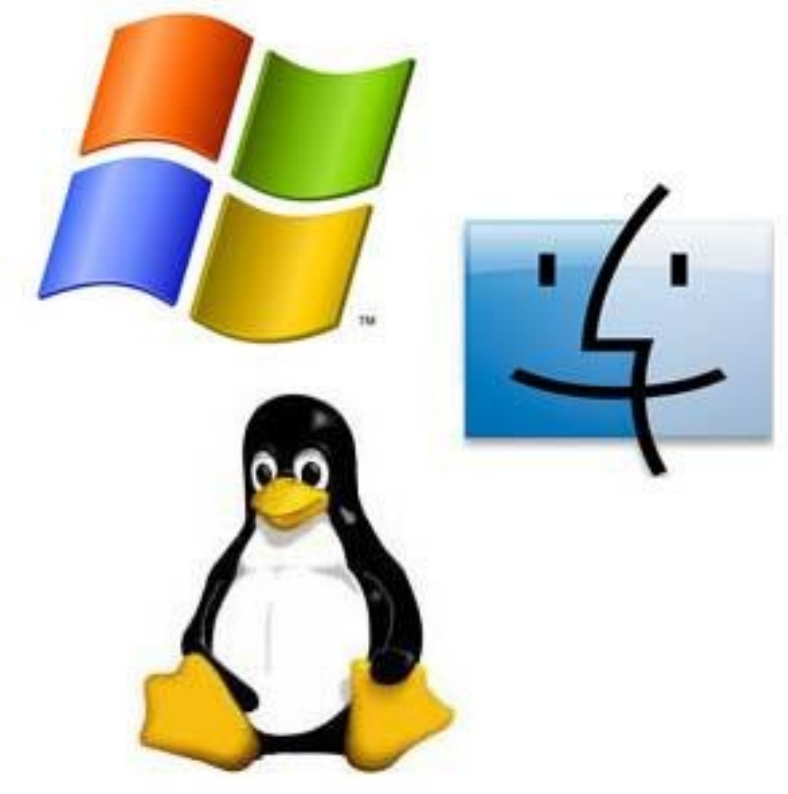


Side note- When do developers trace?

- To better understand their software
- To solve problems
- Typically, by the time a developer traces, they are already in a bad mood.

Tracer Requirements

- Easy to use
- Easy to set-up
- Works in the eclipse environment
- Hopefully already in code
- No lost events



~~Solution 1: JVMTI~~

- Simple
- Interfaces to LTTng in windows and something else in Windows/MacOS… not my problem.
- Code: OnLoad add callbacks for function entry and exit.
- Before JVMTI: 100KEv/s
- After JVMTI: 200Ev/s
- After JVMTI – UST enabled: 199.9999999 Ev/S
- After JVMTI – Chrome : 190ish Ev/s and can’t view in Chrome :(
- https://git.eclipse.org/r/#/c/85932/

~~Solution 2: UST in Java~~

- COMPLICATED
- Hard to maintain
- We only had 1 month to look into this
- Still need to instrument code
- Does not solve the problem

~~Solution 3: ByteCode Instrumentation~~

- Faster than JVMTI
- Way too involved to write

~~Solution 4: Jul~~

- Pretty fast
- Easy to implement
- Problem: events designed in an ad-hoc manner
- Cannot provide out of the box support for everything

Solution: JUL++

- Helpers to make events fit provided models
- Makes adding tracepoints easy
- Define event models:
  - Durations
  - Object Life Spans
  - Asynchronous Flows
  - Samples/Counters
  - Markers
- Should support other loggers (sl4j, log4j, …)
- Inspired by Google's trace event format

Durations

- Call Stack view, Flame Graph view, Function Descriptive Statistics (soon), Call Graph view (soon?)
  - Useful to investigate classic profiling problems
  - Simple API
  - Sub-method precision
  - Handles exceptions

```
try (ScopeLog log = new ScopeLog(LOGGER, Level.FINE, "StateSystem:FullQuery", //$NON-NLS-1$
    "ssid", getSSID(), "ts", t);) { //$NON-NLS-1$ //$NON-NLS-2
    final int nbAttr = getNbAttributes();
    …
}
```

Object Life Spans

- Object Life Spans
  - Great for detecting leaks.
  - Long lived objects and big objects should be tracked
  - Using Timegraph XML view

```
private static final class LivingObject {

    private final @NonNull Logger fLog;

    public LivingObject(@NonNull Logger logger) {
        fLog = logger;
        TraceCompassLogUtils.traceObjectCreation(fLog, Level.FINE, this);
    }

    @Override
    protected void finalize() throws Throwable {
        TraceCompassLogUtils.traceObjectDestruction(fLog, Level.FINE, this);
        super.finalize();
    }
}
```

Asynchronous flow

- Adds causality to call stack view (who triggered the calls)
- Call entry – exit with ID. Can be followed across threads
- Future: stream and scatter gather analysis.

```
TraceCompassLogUtils.traceAsyncStart(logger, Level.FINE, "network connect", "net",
10);
TraceCompassLogUtils.traceAsyncStart(logger, Level.FINER, "network lookup", "net",
10);
TraceCompassLogUtils.traceAsyncNested(logger, Level.FINER, "network cache", "net",
10);
// anon message
TraceCompassLogUtils.traceAsyncStart(logger, Level.FINER, null, null, 0);
TraceCompassLogUtils.traceAsyncEnd(logger, Level.FINER, null, null, 0);
```

Samplers / Counters

- Plotted in a common X line chart
  - Samples→ absolute
  - Counter → change
  - Very useful as either a progress monitor or to find items

Ex: an event is logged every 10000 read items (event/segment/state/...)

```
TraceCompassLogUtils.traceCounter(logger, Level.CONFIG, "levels",
"awesome", 10);
TraceCompassLogUtils.traceCounter(logger, Level.CONFIG, "levels",
"awesome", 20,"cringe", 10);
```

Markers

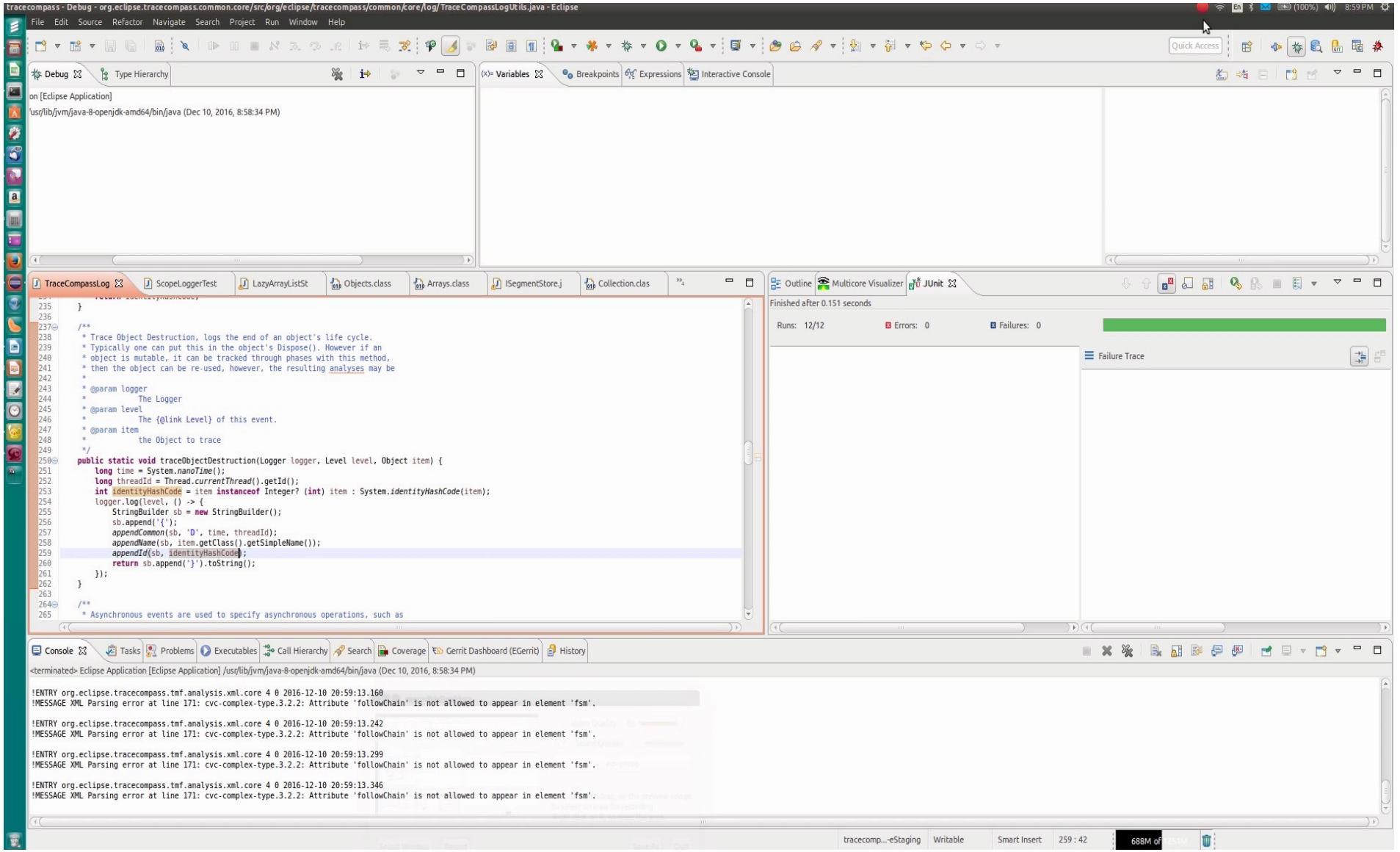
- Time based markers in all views

Decorate views using Eclipse markers

```
TraceCompassLogUtils.traceMarker(logger, Level.CONFIG, "instant", 0);
TraceCompassLogUtils.traceMarker(logger, Level.CONFIG, "colored", 15,
"color", 0xaabccdd);
```

DEMO

Trace Investigation

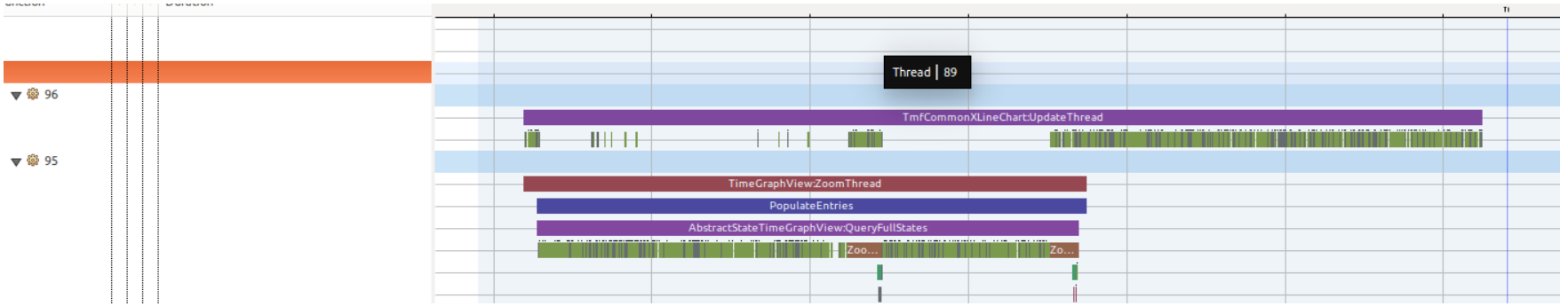


Problem Recap

State system appears to have synchronized full queries.

Many views full query simultaneously (Resources view, Control Flow View, CPU Usage View,...)

Green = Query full state



Recap – From a dev's perspective

- LTTng-UST is great as a transport layer, it is not an application tracer as much as an event collection mechanism.
- We need to provide tracing patterns to allow developers to plug items into their code and have views just work.
- We need to provide a way to launch tracing easily, like Google Chrome does.
- Trace Compass instrumentation already highlighted many issues
- Will bring this to the rest of Eclipse. CDT/Egerrit already interested.

Q And A

Reminders :

Requirements (7)

Java tracing : JVMTI (8) – Java UST (9) – Byte code(10)

JUL (11) – JUL++ (12)

Event Model: Durations (13)

Object Life Spans (14) - Asynchronous Flows (15)

Samples – Counters(16) – Markers (17)

Demo: Aquisition (19) – Analysis (20)

Recaps: Problem (21) – Eclipse Devs Opportunities (22)

REFERENCES

- Project pages
  - http://tracecompass.org
  - http://projects.eclipse.org/projects/tools.tracecompass
  - Is Trace Compass Fast Yet? http://istmffastyet.dorsal.polymtl.ca/
- Documentation
  - Trace Compass User Guide
  - Trace Compass Developer Guide

CONTACTS

- matthew.khouzam@ericsson.com
- Mailing list
  - tracecompass-dev@eclipse.org
- IRC
  - oftc.net #tracecompass
- Mattermost
  - https://mattermost-test.eclipse.org/eclipse/channels/trace-compass

ERICSSON