Using web technologies for trace visualization tools

Yonni CHEN KUANG PIAO

December 7, 2017

École Polytechnique de Montréal
Distributed Open Reliable Systems Analysis Lab
Presentation plan

1 Previously
2 Contribution to Trace Compass
3 What are the possibilities now?
4 In practice : TracEscape
5 What’s next?
Previously
Contribution to Trace Compass

Before

```
+----------------------------------+
| SWT View                         |
+----------------------------------+
| Viewer                           |
+----------------------------------+
| Analysis                         |
+----------------------------------+
| State System                     |
```

After

```
+----------------------------------+
| SWT View                         |
+----------------------------------+
| Viewer                           |
+----------------------------------+
| Data providers                   |
+----------------------------------+
| Analysis                         |
+----------------------------------+
| State System                     |
```
Contribution to Trace Compass

- There is less business logic in the UI layer
- Data providers return a “ready to render” model
  - Simple
  - Immutable
  - Serializable
- All XY views are refactored
  - Patches for Control Flow and Call stack are ready to merge
  - Patch for Event table is on Gerrit
- Easier to test
Contribution to Trace Compass
What are the possibilities now? 

- Java View?
- C++ View?
- JavaScript View?

Trace Analysis Server Protocol

Trace Compass Server

REST API

Data providers

Analysis

State System
What are the possibilities now?
What are the possibilities now?

Imagine if we have the possibility to...

• Switch to a C++ backend and keep Trace Compass Eclipse?
• Visualize traces without Eclipse with command line?
• Visualize traces which are not on your filesystem?
• Visualize very large traces in Chromium?
• Have a Theia plugin to visualize traces?
In practice : TracEscape
## In practice: TracEscape

### Data exchange

<table>
<thead>
<tr>
<th>Trace size (in MB)</th>
<th>JSON</th>
<th>Protobuf</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>346 KB</td>
<td>85.8 KB</td>
<td>-75.2 %</td>
</tr>
<tr>
<td>57.8</td>
<td>87.8 KB</td>
<td>28.5 KB</td>
<td>-67.5 %</td>
</tr>
<tr>
<td>793</td>
<td>6.9 MB</td>
<td>1.9 MB</td>
<td>-72.5%</td>
</tr>
<tr>
<td>2200</td>
<td>17.3 MB</td>
<td>4.6 MB</td>
<td>-73.9%</td>
</tr>
</tbody>
</table>
What’s next?
What’s next?

✓ We can trace TracEscape with Chromium Developer tools
✓ We can trace TraceCompass
✓ We defined a first draft of the Trace Analysis Server Protocol

➢ Performance benchmarks BEFORE/AFTER data providers
➢ Merge the two trace and see what’s happening
➢ Automatic benchmarks
➢ Investigating gRPC vs HTTP REST
Questions?

yonni.scholars@gmail.com

https://github.com/cheninator/trace-scape