



**POLYTECHNIQUE  
MONTREAL**

TECHNOLOGICAL  
UNIVERSITY

# A New Flexible Architecture for Trace Compass

**Ericsson Canada TC team: Simon Delisle, Bernd Hufmann, Matthew Khouzam, Patrick Tasse**  
**Polytechnique: Geneviève Bastien, Michel Dagenais (presenter)**

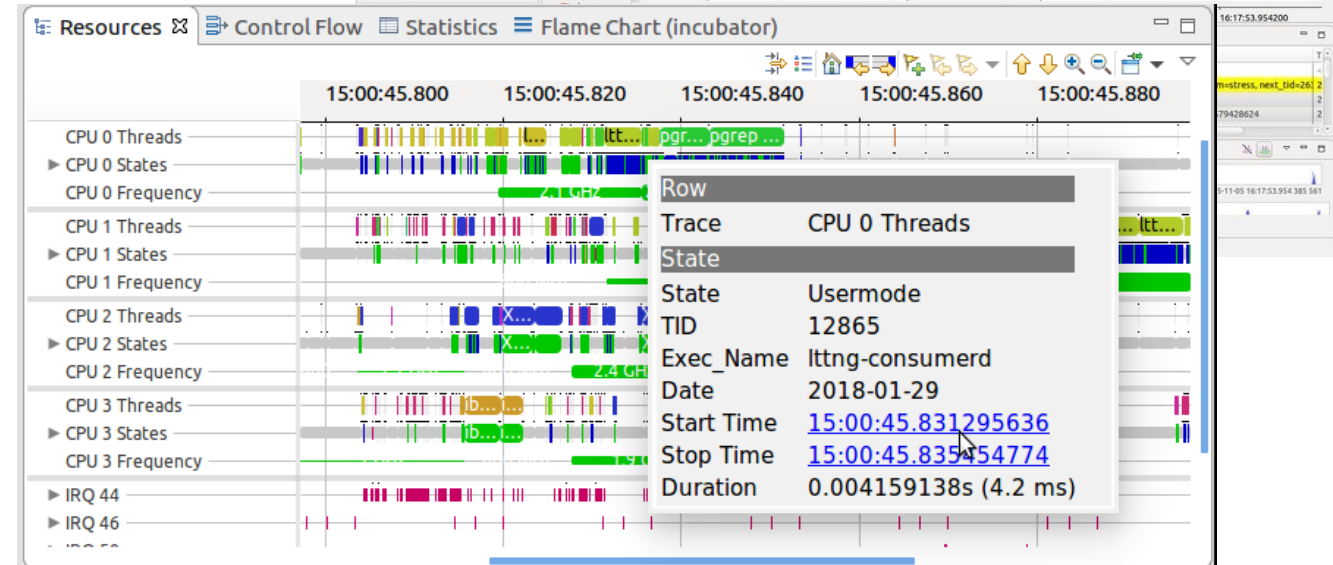
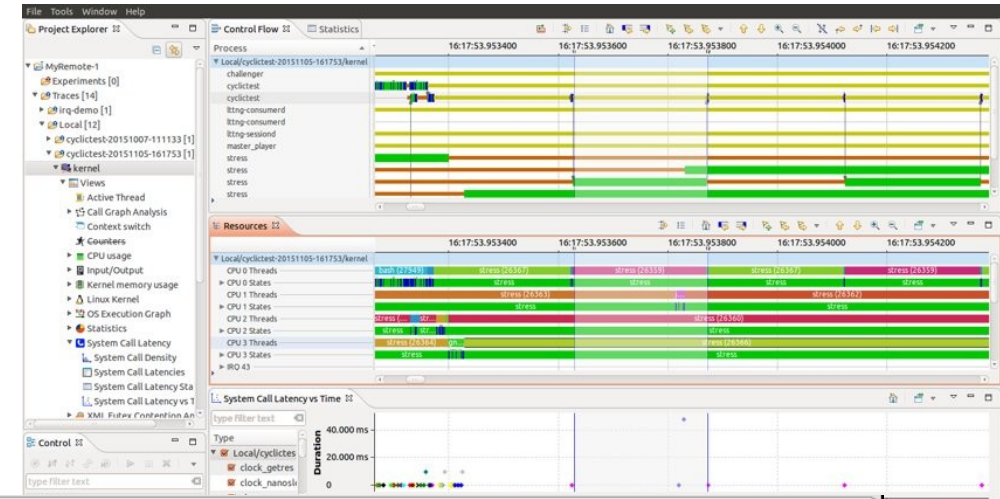
# Summary

- What is Trace Compass and Theia
- Trace Compass evolution
- Trace Server Protocol
- Trace Compass on Theia
- Scripting with Ease
- Conclusion

Some of this work was performed as part of a Collaborative Research and Development project at Polytechnique Montreal with Ericsson, Ciena and EfficiOS, with funding from NSERC, Prompt, Ericsson, Ciena, Google and EfficiOS.

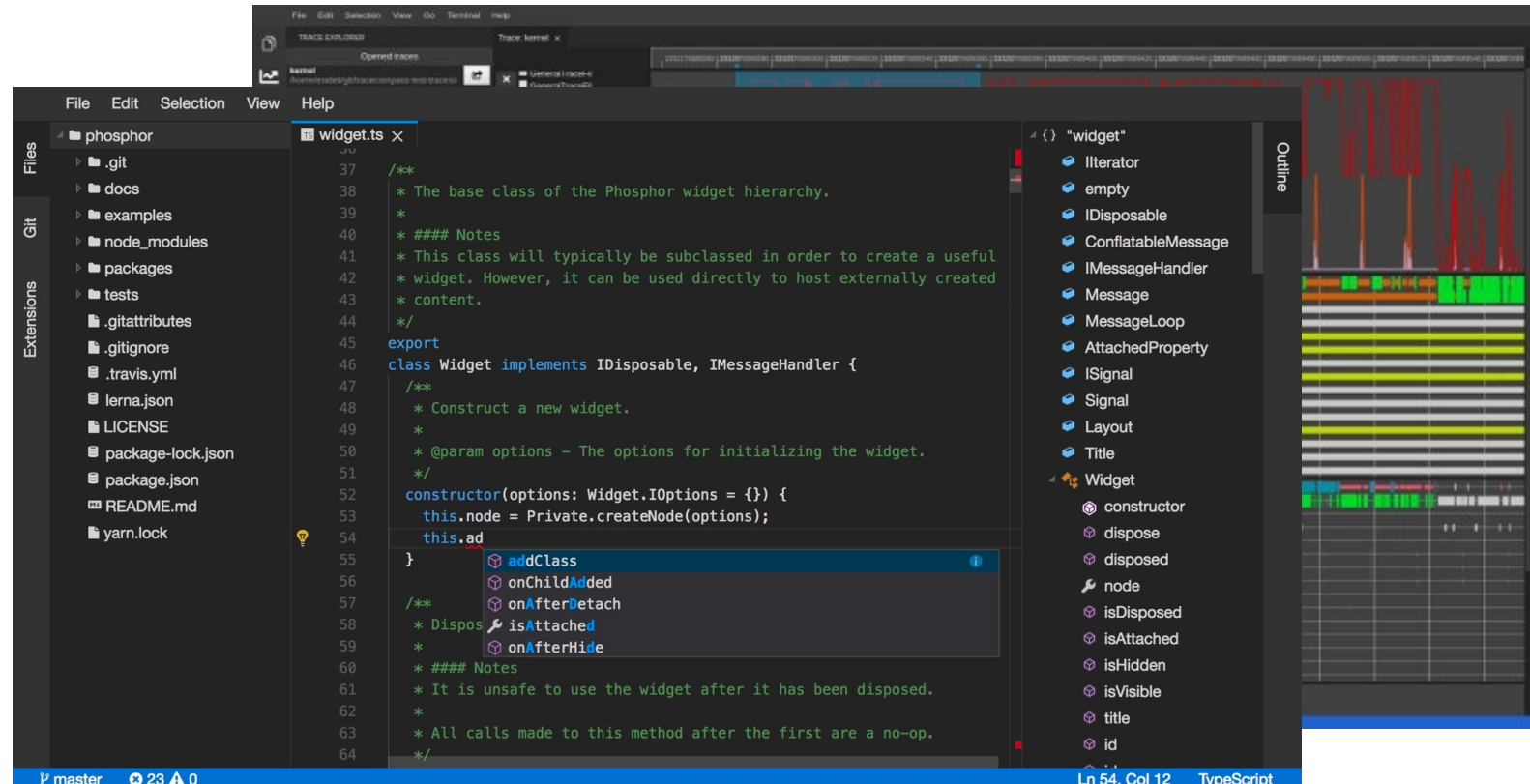
# Trace Compass: an open source trace analysis tool to solve performance and reliability problems

- Trace
  - Series of events over time
  - Event collected at tracepoints during program execution
  - Each event has a type and payload
- Use the events as input for analysis
- Create visualization graphs with these analysis
- Tracing use cases
  - Profile application
  - Find long executions
  - Investigate real-time deadlines
  - Find memory or load issues
  - Investigate concurrency problems



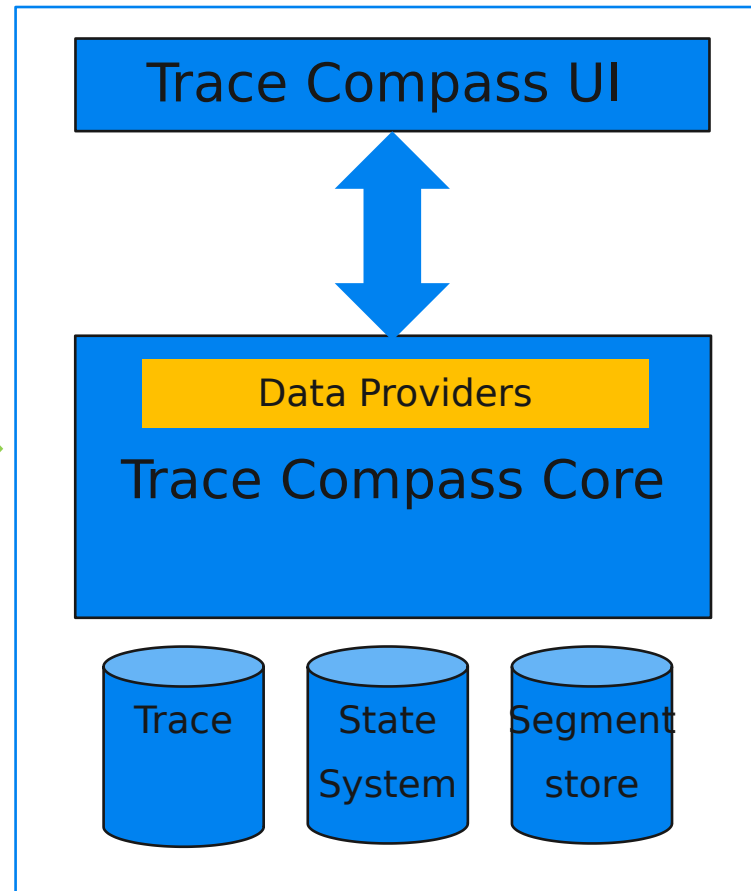
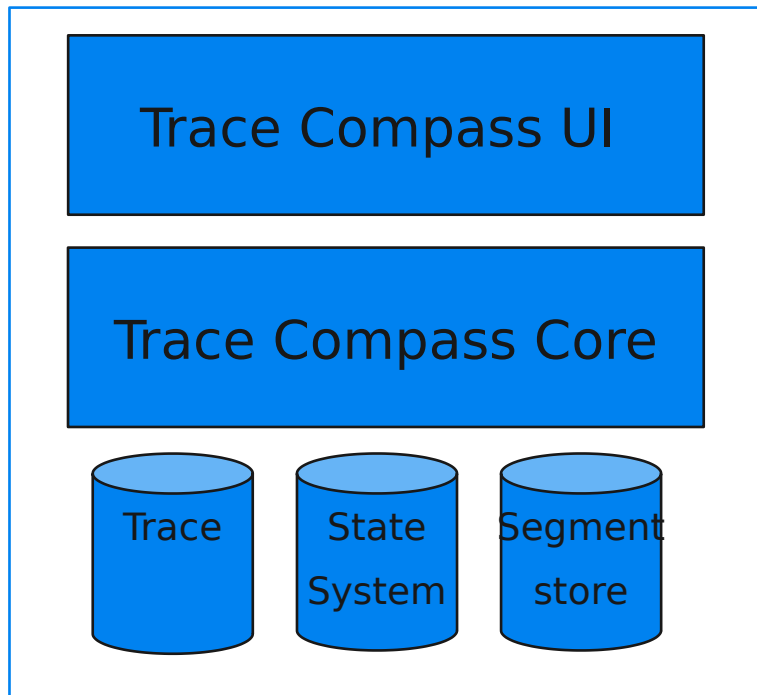
# Theia: an extensible open-source framework to develop multi-language IDEs for the cloud and desktop using state-of-the-art web technologies

- Cloud and desktop IDE
- Modules in different languages accessed through protocols.
- Based on several existing state-of-the-art modules :
  - Monaco editor
  - Chromium
  - React.js
  - Language servers
  - Debug adapters
  - Visual Studio Code extensions

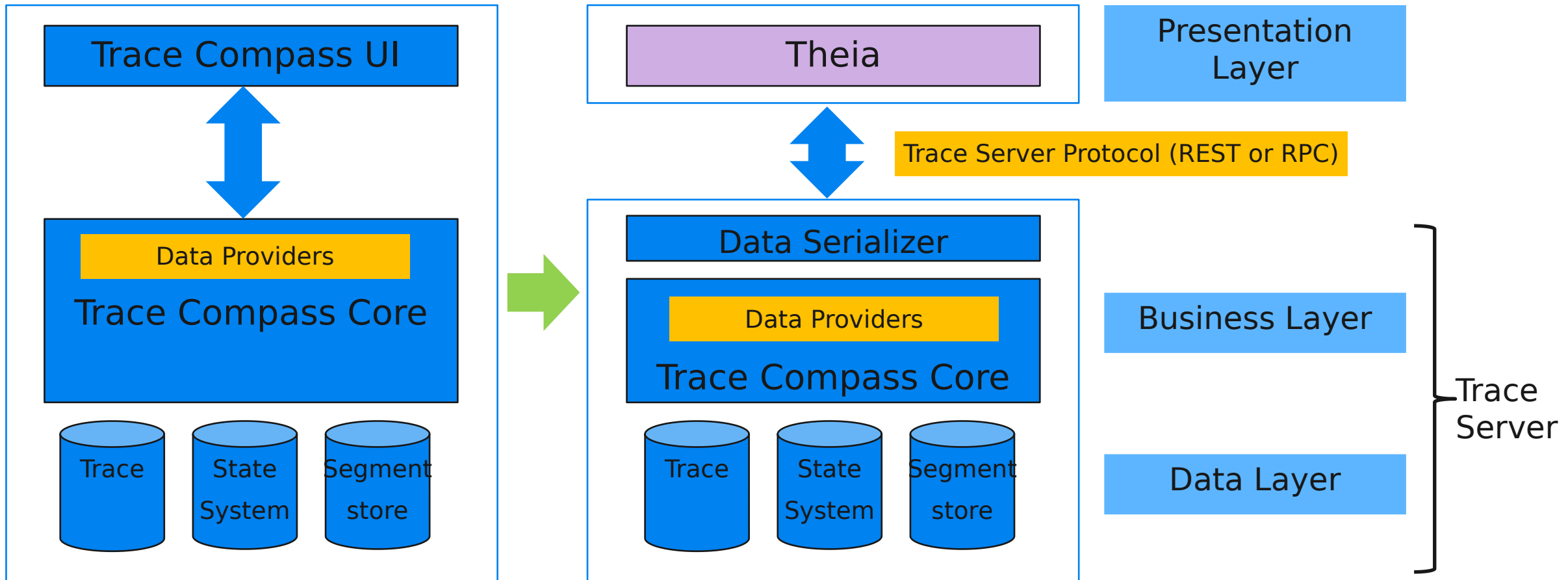


# Trace Compass architecture

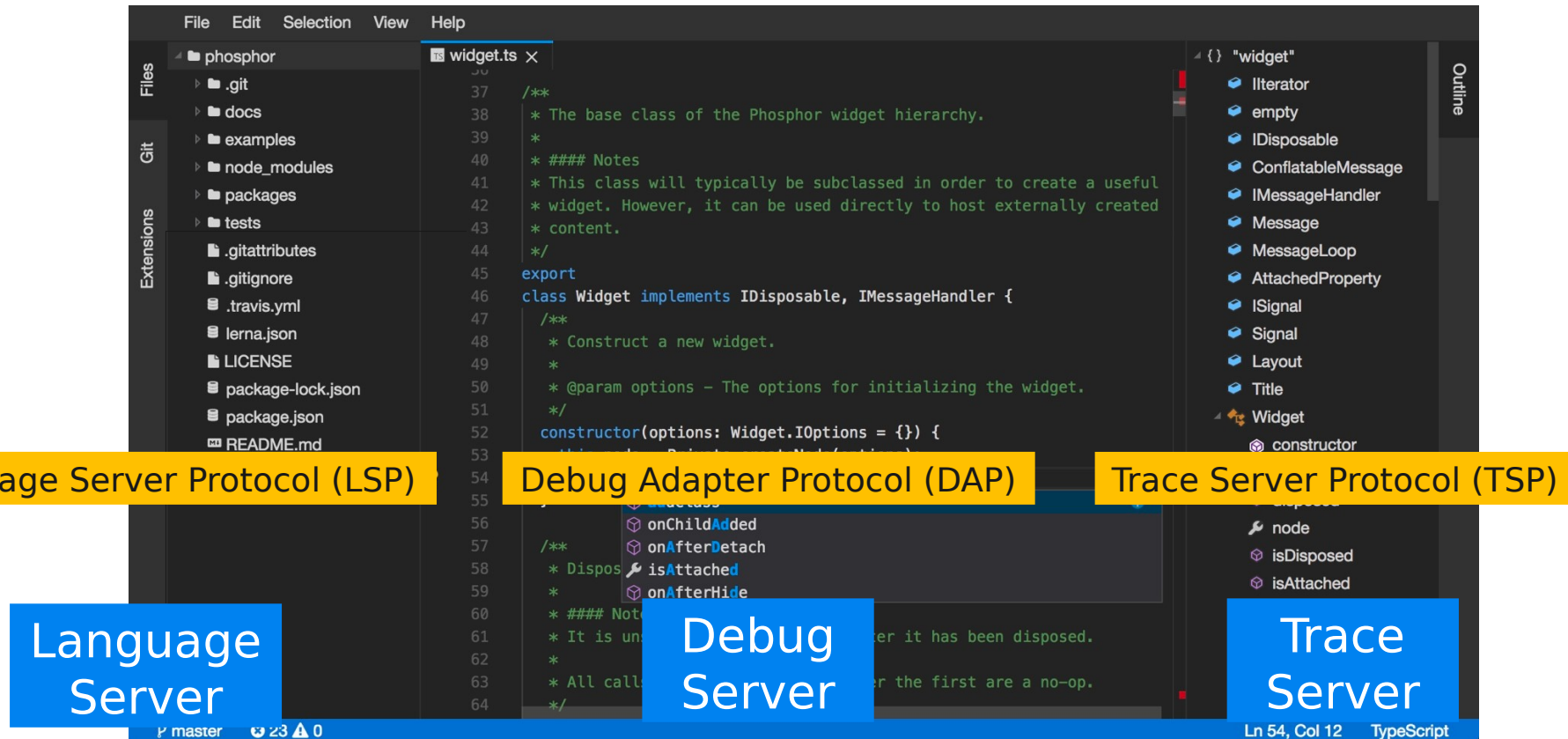
Current (ongoing)



# Proposed Client-Server architecture



# Trace Server Protocol



The screenshot shows a VS Code editor with a TypeScript file named `widget.ts`. The code defines a `Widget` class that implements `IDisposable` and `IMessageHandler`. The class has a constructor and several methods including `onChildAdded`, `onAfterDetach`, `onAttached`, and `onAfterHide`. The code is annotated with JSDoc comments.

Overlaid on the screenshot are three yellow boxes representing protocols and three blue boxes representing servers:

- Language Server Protocol (LSP)** (yellow box) is associated with the **Language Server** (blue box).
- Debug Adapter Protocol (DAP)** (yellow box) is associated with the **Debug Server** (blue box).
- Trace Server Protocol (TSP)** (yellow box) is associated with the **Trace Server** (blue box).

The status bar at the bottom indicates the current file is `Ln 54, Col 12` and the language is `TypeScript`.

# Trace Server Protocol (TSP)

- Protocol built to handle communication between backend and frontend of trace viewer, allowing traces to reside and be analysed on the backend.
- Exchange visualization data between a client and a server
- Trace management
- Server-side filtering and searching
- <https://github.com/theia-ide/trace-server-protocol>
- Integration with Theia using tsp-typescript-client
  - TSP ready client to perform your requests
  - Abstract the technology used (REST, HTTP)
  - NPM package available
  - <https://github.com/theia-ide/tsp-typescript-client>
- Contributions and feedback are welcome



# Opportunities

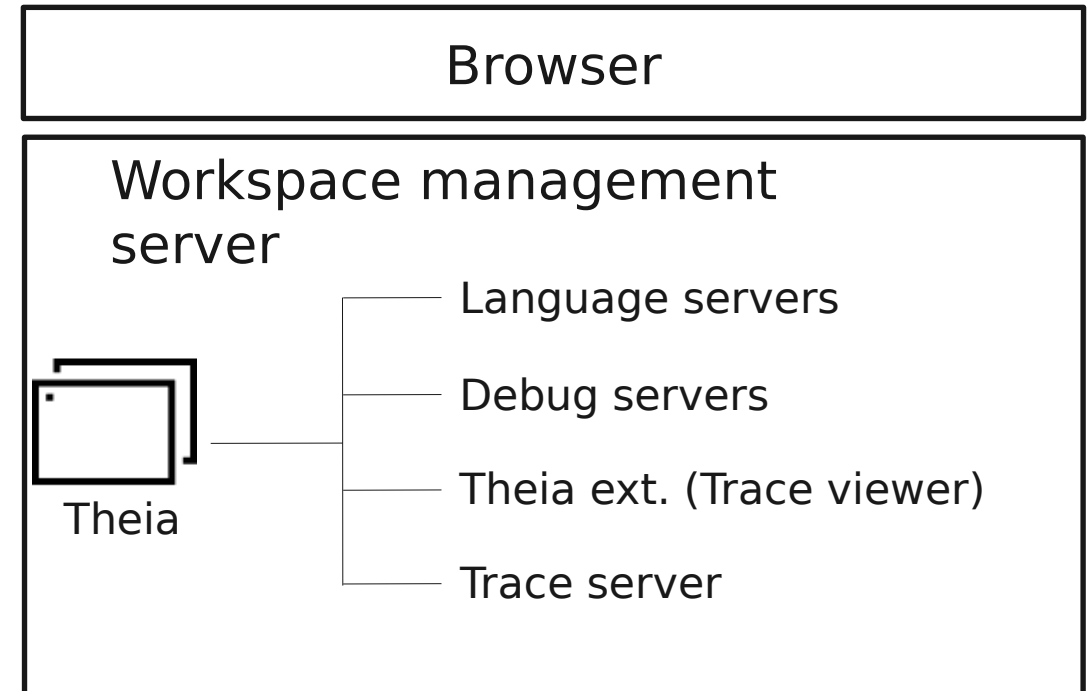
- Modular architecture (using modules in different languages leveraging LSP, DAP...).
- Thin UI client or scripted access.
- Leveraging modern UI technologies (React.js)
- Continuous integration (e.g. traces directly from Jenkins)
- Integration with bug report tools (e.g. open traces)
- Integration with workspace management (e.g. Eclipse Che)
- Higher scalability / Performance
- Security (traces in the cloud)

# Leveraging LSP and DAP

CPU	Event type	Contents	
<srch>	<srch>	<srch>	
2	lttnng_ust_cyg_profile_fast:func_entry	addr=0x402b80, context._vpid=26070, context._vtid=26070, context._procname=	<div> <p>— DAP to get file and line number</p> <p>— Then use LSP to lookup source code</p> </div>
2	lttnng_ust_cyg_profile_fast:func_exit	context._vpid=26070, context._vtid=26070, context._procname=sort	
2	0	ust_sequence:ENTER	
	0	ust_sequence:INFO	
		file=simple_server_main.cpp, line=97, content=messageHandler()	
		file=simple_server_main.cpp, line=113, content=player player1	
2	lttnng_ust_cyg_profile_fast:func_exit	context._vpid=26070, context._vtid=26070, context._procname=sort	
0	kmem_kmalloc	call_site=0xfffffffffa08b989	<div> <p>LSP to lookup source code</p> </div>
		bytes_alloc=	

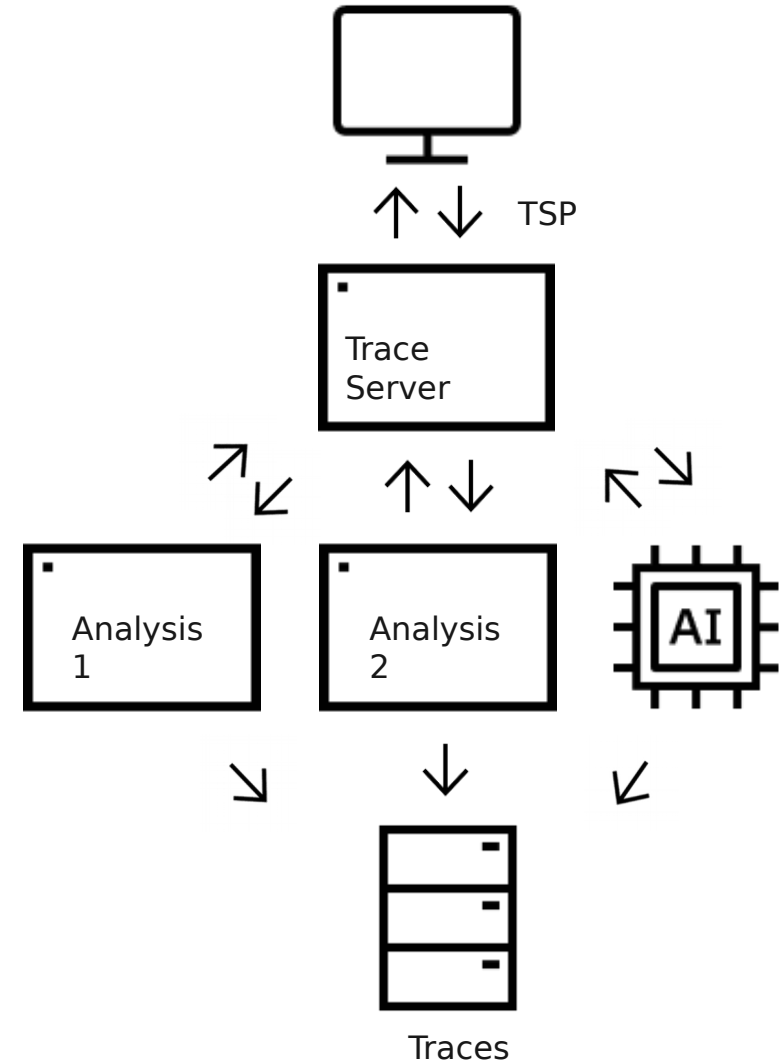
# Integration with workspace management

- Prepare workspace for trouble-shooting sessions
  - Cloud IDE
    - Get source code
      - LSP
    - Setup debuggers
      - DAP
    - Setup trace viewer
      - TSP
  - Share trouble-shooting sessions (workspaces)



# Higher Scalability

- Enables micro-services
- Distributed architecture
- Parallel, distributed analysis
  - Different traces
  - Same traces, different analysis
- Analyze traces that exceed local disk space



- 
- The screenshot displays the Perfetto Trace Explorer application. The interface is divided into several sections:
- Top Bar:** Contains standard application menus (File, Edit, Selection, View, Go, Terminal, Help).
  - Left Sidebar:**
    - Opened traces:** Lists the loaded trace file, 'kernel.x'.
    - Available analysis:** A list of analysis tools with checkboxes, including Thread States, Events Table, Disk IO View, Histogram, Memory Usage, Resource Status, and Time Graph Toolkit.
    - Thread States:** A detailed view of thread states, showing a list of threads (e.g., kworker/0/0, krefill, kworker/0/1, kworker/0/2, kworker/0/3, kworker/0/4, kworker/0/5, kworker/0/6, kworker/0/7, kworker/0/8, kworker/0/9, kworker/0/10, kworker/0/11, kworker/0/12, kworker/0/13, kworker/0/14, kworker/0/15, kworker/0/16, kworker/0/17, kworker/0/18, kworker/0/19, kworker/0/20, kworker/0/21, kworker/0/22, kworker/0/23, kworker/0/24, kworker/0/25, kworker/0/26, kworker/0/27, kworker/0/28, kworker/0/29, kworker/0/30, kworker/0/31, kworker/0/32, kworker/0/33, kworker/0/34, kworker/0/35, kworker/0/36, kworker/0/37, kworker/0/38, kworker/0/39, kworker/0/40, kworker/0/41, kworker/0/42, kworker/0/43, kworker/0/44, kworker/0/45, kworker/0/46, kworker/0/47, kworker/0/48, kworker/0/49, kworker/0/50, kworker/0/51, kworker/0/52, kworker/0/53, kworker/0/54, kworker/0/55, kworker/0/56, kworker/0/57, kworker/0/58, kworker/0/59, kworker/0/60, kworker/0/61, kworker/0/62, kworker/0/63, kworker/0/64, kworker/0/65, kworker/0/66, kworker/0/67, kworker/0/68, kworker/0/69, kworker/0/70, kworker/0/71, kworker/0/72, kworker/0/73, kworker/0/74, kworker/0/75, kworker/0/76, kworker/0/77, kworker/0/78, kworker/0/79, kworker/0/80, kworker/0/81, kworker/0/82, kworker/0/83, kworker/0/84, kworker/0/85, kworker/0/86, kworker/0/87, kworker/0/88, kworker/0/89, kworker/0/90, kworker/0/91, kworker/0/92, kworker/0/93, kworker/0/94, kworker/0/95, kworker/0/96, kworker/0/97, kworker/0/98, kworker/0/99, kworker/0/100, kworker/0/101, kworker/0/102, kworker/0/103, kworker/0/104, kworker/0/105, kworker/0/106, kworker/0/107, kworker/0/108, kworker/0/109, kworker/0/110, kworker/0/111, kworker/0/112, kworker/0/113, kworker/0/114, kworker/0/115, kworker/0/116, kworker/0/117, kworker/0/118, kworker/0/119, kworker/0/120, kworker/0/121, kworker/0/122, kworker/0/123, kworker/0/124, kworker/0/125, kworker/0/126, kworker/0/127, kworker/0/128, kworker/0/129, kworker/0/130, kworker/0/131, kworker/0/132, kworker/0/133, kworker/0/134, kworker/0/135, kworker/0/136, kworker/0/137, kworker/0/138, kworker/0/139, kworker/0/140, kworker/0/141, kworker/0/142, kworker/0/143, kworker/0/144, kworker/0/145, kworker/0/146, kworker/0/147, kworker/0/148, kworker/0/149, kworker/0/150, kworker/0/151, kworker/0/152, kworker/0/153, kworker/0/154, kworker/0/155, kworker/0/156, kworker/0/157, kworker/0/158, kworker/0/159, kworker/0/160, kworker/0/161, kworker/0/162, kworker/0/163, kworker/0/164, kworker/0/165, kworker/0/166, kworker/0/167, kworker/0/168, kworker/0/169, kworker/0/170, kworker/0/171, kworker/0/172, kworker/0/173, kworker/0/174, kworker/0/175, kworker/0/176, kworker/0/177, kworker/0/178, kworker/0/179, kworker/0/180, kworker/0/181, kworker/0/182, kworker/0/183, kworker/0/184, kworker/0/185, kworker/0/186, kworker/0/187, kworker/0/188, kworker/0/189, kworker/0/190, kworker/0/191, kworker/0/192, kworker/0/193, kworker/0/194, kworker/0/195, kworker/0/196, kworker/0/197, kworker/0/198, kworker/0/199, kworker/0/200, kworker/0/201, kworker/0/202, kworker/0/203, kworker/0/204, kworker/0/205, kworker/0/206, kworker/0/207, kworker/0/208, kworker/0/209, kworker/0/210, kworker/0/211, kworker/0/212, kworker/0/213, kworker/0/214, kworker/0/215, kworker/0/216, kworker/0/217, kworker/0/218, kworker/0/219, kworker/0/220, kworker/0/221, kworker/0/222, kworker/0/223, kworker/0/224, kworker/0/225, kworker/0/226, kworker/0/227, kworker/0/228, kworker/0/229, kworker/0/230, kworker/0/231, kworker/0/232, kworker/0/233, kworker/0/234, kworker/0/235, kworker/0/236, kworker/0/237, kworker/0/238, kworker/0/239, kworker/0/240, kworker/0/241, kworker/0/242, kworker/0/243, kworker/0/244, kworker/0/245, kworker/0/246, kworker/0/247, kworker/0/248, kworker/0/249, kworker/0/250, kworker/0/251, kworker/0/252, kworker/0/253, kworker/0/254, kworker/0/255, kworker/0/256, kworker/0/257, kworker/0/258, kworker/0/259, kworker/0/260, kworker/0/261, kworker/0/262, kworker/0/263, kworker/0/264, kworker/0/265, kworker/0/266, kworker/0/267, kworker/0/268, kworker/0/269, kworker/0/270, kworker/0/271, kworker/0/272, kworker/0/273, kworker/0/274, kworker/0/275, kworker/0/276, kworker/0/277, kworker/0/278, kworker/0/279, kworker/0/280, kworker/0/281, kworker/0/282, kworker/0/283, kworker/0/284, kworker/0/285, kworker/0/286, kworker/0/287, kworker/0/288, kworker/0/289, kworker/0/290, kworker/0/291, kworker/0/292, kworker/0/293, kworker/0/294, kworker/0/295, kworker/0/296, kworker/0/297, kworker/0/298, kworker/0/299, kworker/0/300, kworker/0/301, kworker/0/302, kworker/0/303, kworker/0/304, kworker/0/305, kworker/0/306, kworker/0/307, kworker/0/308, kworker/0/309, kworker/0/310, kworker/0/311, kworker/0/312, kworker/0/313, kworker/0/314, kworker/0/315, kworker/0/316, kworker/0/317, kworker/0/318, kworker/0/319, kworker/0/320, kworker/0/321, kworker/0/322, kworker/0/323, kworker/0/324, kworker/0/325, kworker/0/326, kworker/0/327, kworker/0/328, kworker/0/329, kworker/0/330, kworker/0/331, kworker/0/332, kworker/0/333, kworker/0/334, kworker/0/335, kworker/0/336, kworker/0/337, kworker/0/338, kworker/0/339, kworker/0/340, kworker/0/341, kworker/0/342, kworker/0/343, kworker/0/344, kworker/0/345, kworker/0/346, kworker/0/347, kworker/0/348, kworker/0/349, kworker/0/350, kworker/0/351, kworker/0/352, kworker/0/353, kworker/0/354, kworker/0/355, kworker/0/356, kworker/0/357, kworker/0/358, kworker/0/359, kworker/0/360, kworker/0/361, kworker/0/362, kworker/0/363, kworker/0/364, kworker/0/365, kworker/0/366, kworker/0/367, kworker/0/368, kworker/0/369, kworker/0/370, kworker/0/371, kworker/0/372, kworker/0/373, kworker/0/374, kworker/0/375, kworker/0/376, kworker/0/377, kworker/0/378, kworker/0/379, kworker/0/380, kworker/0/381, kworker/0/382, kworker/0/383, kworker/0/384, kworker/0/385, kworker/0/386, kworker/0/387, kworker/0/388, kworker/0/389, kworker/0/390, kworker/0/391, kworker/0/392, kworker/0/393, kworker/0/394, kworker/0/395, kworker/0/396, kworker/0/397, kworker/0/398, kworker/0/399, kworker/0/400, kworker/0/401, kworker/0/402, kworker/0/403, kworker/0/404, kworker/0/405, kworker/0/406, kworker/0/407, kworker/0/408, kworker/0/409, kworker/0/410, kworker/0/411, kworker/0/412, kworker/0/413, kworker/0/414, kworker/0/415, kworker/0/416, kworker/0/417, kworker/0/418, kworker/0/419, kworker/0/420, kworker/0/421, kworker/0/422, kworker/0/423, kworker/0/424, kworker/0/425, kworker/0/426, kworker/0/427, kworker/0/428, kworker/0/429, kworker/0/430, kworker/0/431, kworker/0/432, kworker/0/433, kworker/0/434, k

# Scripted Analyses with Ease

- Finite number of available analyses
- Some flexibility with XML analyses:
  - Very verbose
  - Hard to read
  - Hard to debug
  - But it works!
- Ultimate flexibility: scripting

```
</stateChange>
<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>

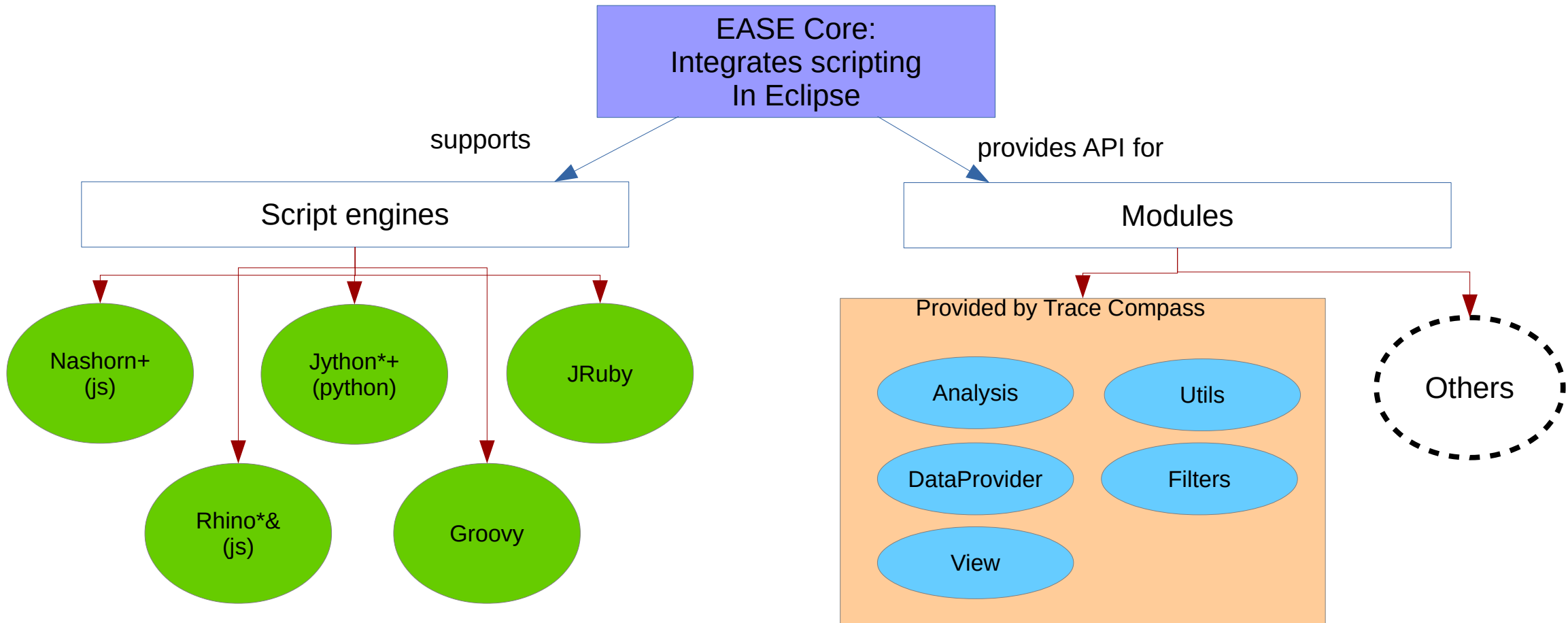
<action id="push_event_type">
  <!-- Push the current event to the thread's callstack -->
  <stateChange>
    <stateAttribute type="location" value="CurrentThread" />
    <stateAttribute type="constant" value="CallStack" />
    <stateValue type="eventField" value="evName" stack="push"/>
  </stateChange>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

# EASE : Eclipse Advanced Scripting Environment



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

# Conclusion

- Functionality in Trace Compass migrated gradually to Data Providers
- Most new features are implemented in the backend and work on both frontends
- New views are added to Theia Trace Compass gradually
- Feature parity will not be reached for at least several months
- Some experimental views may be implemented in Theia first
- A new IDE for the Cloud with Theia and Trace Compass



# Reaching us

- Trace Compass: <http://tracecompass.org>
- Mailing list: [tracecompass-dev@eclipse.org](mailto:tracecompass-dev@eclipse.org)
- IRC: oftc.net #tracecompass
- Trace Server Protocol
  - <https://github.com/theia-ide/trace-server-protocol>
  - <https://github.com/theia-ide/tsp-typescript-client>
- Theia frontend prototype
  - <https://github.com/delislesim/theia-trace-extension/tree/theiaCompass>
- Trace Compass scripting demo: <http://versatic.net/tracingSummit2019.html>