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)

Trace Compass UI

Data Providers

Trace Compass Core

Trace
State System
Segment store
Proposed Client-Server architecture

- Trace Compass Core
  - Data Providers
  - Trace
  - State
  - System
  - Segment
  - store

- Trace Server Protocol (REST or RPC)
- Data Serializer
  - Data Providers
  - Trace
  - State
  - System
  - Segment
  - store

- Theia

- Presentation Layer
- Business Layer
- Data Layer
Trace Server Protocol

Language Server Protocol (LSP)
Debug Adapter Protocol (DAP)
Trace Server Protocol (TSP)

Language Server
Debug Server
Trace Server
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

- DAP to get file and line number
- Then use LSP to lookup source code
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)

Browser

Workspace management server
  • Language servers
  • Debug servers
  • Theia ext. (Trace viewer)
  • Trace server

Theia
Higher Scalability

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

- Theia based prototype using the TSP
- Prototype available on GitHub https://github.com/delislesim/theia-trace-extension/tree/theiaCompass
- Opportunity for a new UI/UX
- React
- Chart.js
- agGrid
- New time graph library
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
EASE : Eclipse Advanced Scripting Environment

EASE Core:
Integrates scripting
In Eclipse

Script engines

- Nashorn+ (js)
- Jython++ (python)
- JRuby
- Rhino*& (js)
- Groovy

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

Modules

- Analysis
- Utils
- DataProvider
- Filters
- View

Provided by Trace Compass

Others
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
• 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