

Tracing Summit 2017

Babeltrace 2

Tailor-made trace analyses

Presenter



Jérémie Galarneau



EfficiOS Inc.

- Vice President
- <http://www.efficios.com>



Maintainer of

- LTTng-Tools
- Babeltrace

Babeltrace 1.x

- MIT licensed Common Trace Format (CTF) reference implementation (2011)
 - Served as the *de facto* LTTng command line trace reader
- Introduced Python bindings to read traces (2013)
 - Provide users a way to prototype an analysis rapidly without using text-based tools (awk, sed, grep, etc.)
 - Debugging
 - Testing
 - Scripts maintained as internal tools by some users
 - Basis of LTTng-analyses
- Provides a CTF production library (CTF-Writer) (2014)
 - Used by perf to convert traces to CTF

Limitations of Babeltrace 1.x

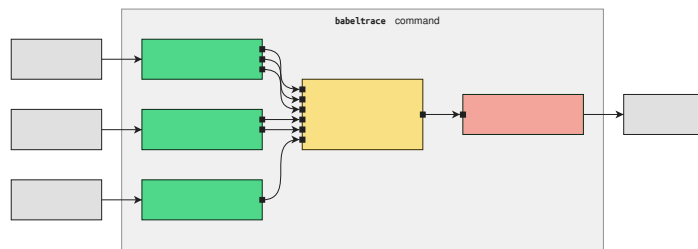
- Only supports in-tree plug-in
 - Does not expose a stable ABI (for plug-ins)
 - No solution to support proprietary (or niche) trace formats out of tree
- Design made it impossible to implement “lttng-live” support as a plug-in
- API quirks
 - Does not allow caching of events,
 - One iterator per trace,
 - Hard/impossible to work with multiple clock sources

Redefining Babeltrace's scope

- Realized that trace analysis tools were not keeping up with the tracers
 - Capturing trace is only half of the battle
 - Use in production shows that working with huge traces is challenging
 - Text-based analyses do not scale and are hard to maintain... but they are useful!
- Project scope changed from a trace *converter* to a trace *manipulation* tool
 - Support more input and output formats,
 - Trim, filter, and add information to traces,
 - Make it easy to assemble “blocks” to build a custom trace analysis pipeline.

Redefining Babeltrace's scope

- Babeltrace becomes a “host” application for trace processing graphs



- User-defined processing graphs
 - Standardize trace processing (shareable graph configurations)
 - Can be assembled programatically
 - Usable from external tools, such as viewers
 - Can be used to process trace “chunks” independently

Building a graph

- Building a graph on the command line

```
$ babeltrace run
```

```
--component=my_source:src.plugin_name.my-src
```

```
--component=my_sink:sink.plugin.my-sink
```

```
--connect=my_source:my_sink
```

Don't worry, there are helpers for common scenarios...

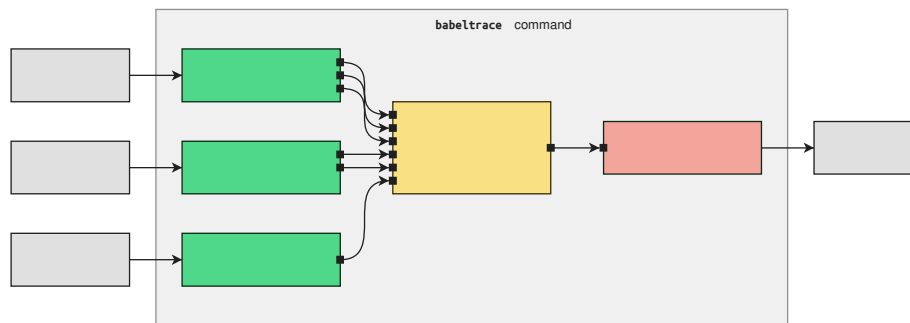
- Graphs can be built using the C and Python APIs

Redefining Babeltrace's scope

- Cross-platform
 - Linux
 - Windows (native and Cygwin)
 - Solaris
 - BSDs
 - macOS
- Preserve the current Babeltrace 1.x Python and CTF-Writer interface

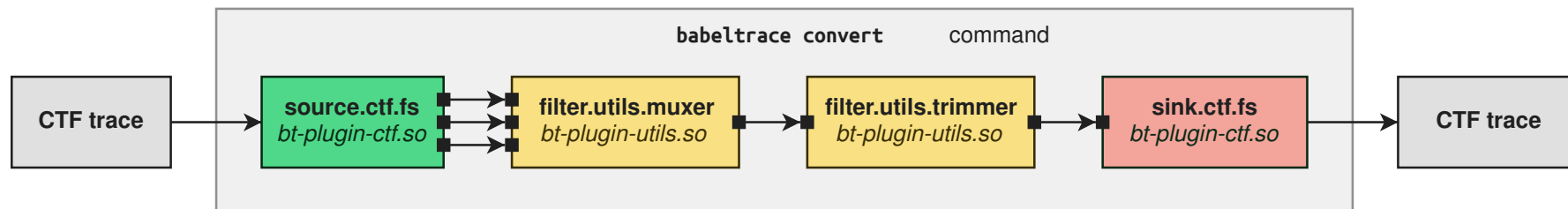
Babeltrace 2.0

- Provides components which allow everything Babeltrace 1.x could do
 - CTF source (reader)
 - CTF sink (writer)
 - LTTng-live source
 - dmesg source
 - Muxer
 - Trimmer
 - Debug info
- Components can be written in C, C++ and Python
 - Stable ABI allows out-of-tree components



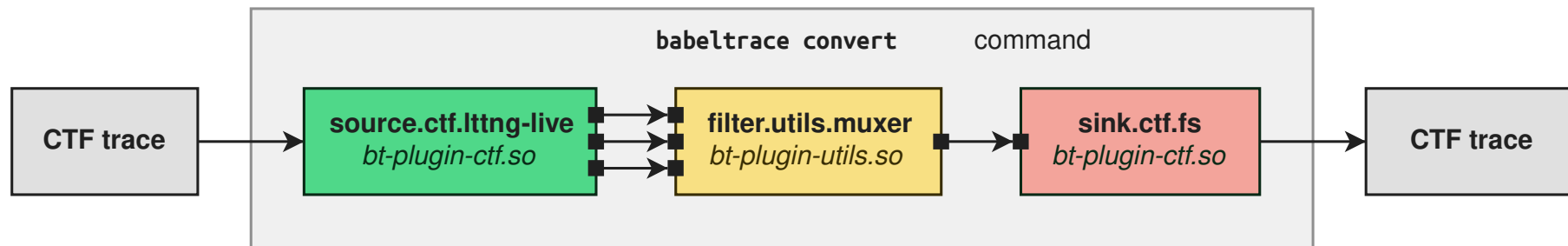
Scenarios – Trimming a trace

```
$ babeltrace /path/to/trace --begin=22:14:38  
--end=22:15:07
```



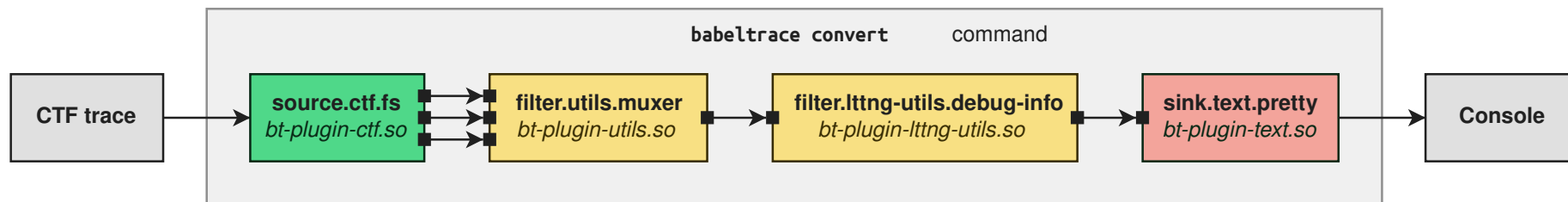
Scenarios – Record part of a live trace

```
$ babeltrace --input-format=lttng-live  
net://localhost/host/my_host/my_session  
--begin=22:05:00 --end=22:06:00
```



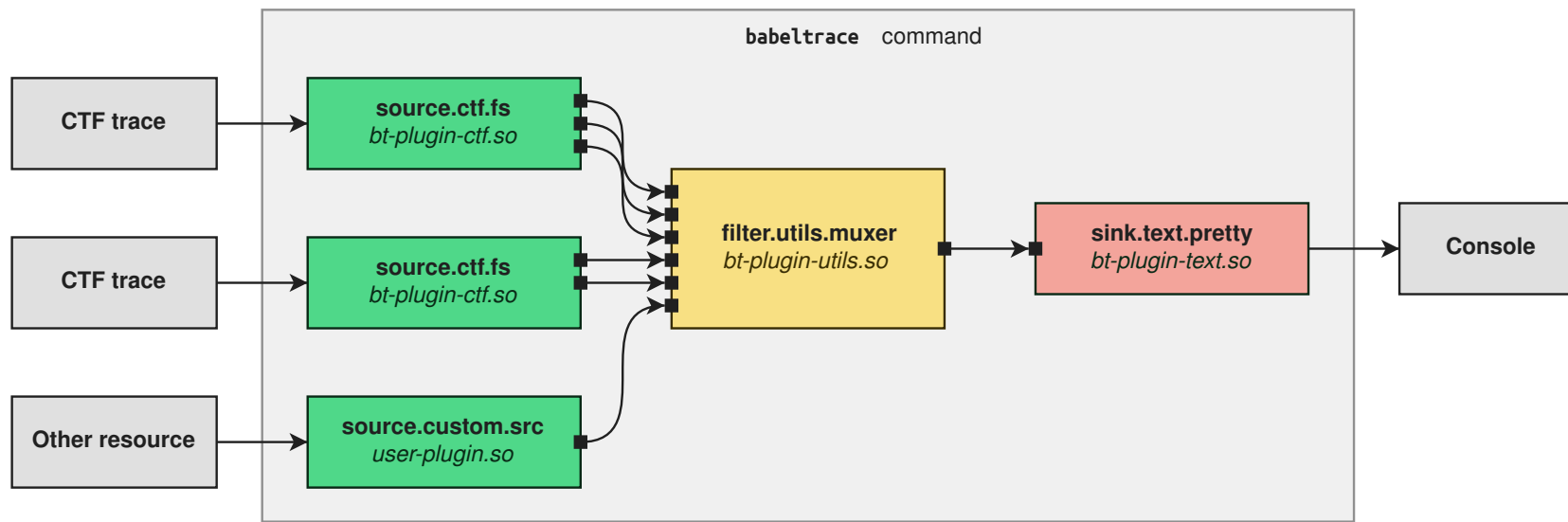
Scenarios – Print debug info

```
$ babeltrace --debug-info /path/to/trace
```



Scenarios – Mux multiple formats

```
$ babeltrace  
  --component source.ctf.fs --path /path/to/trace  
  --component source.text.dmesg --path /tmp/my_dmesg.log
```

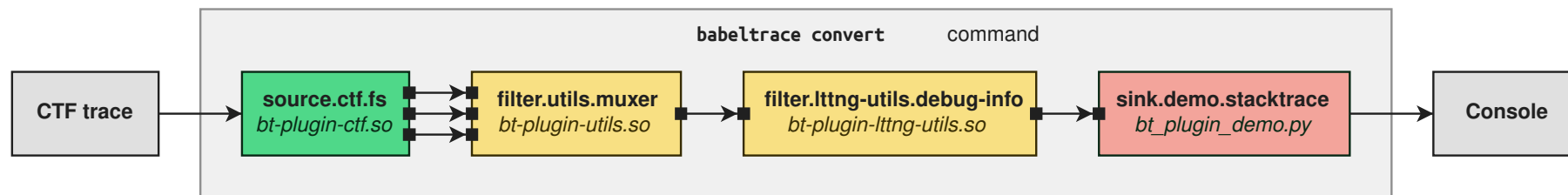


Easy to prototype new components

- Components can be written in Python and used from the command line
 - Insert only the logic you need to build your analysis
- Quick example: a callstack view inspired by *uftrace* in ~50 lines
 - <https://github.com/jgalar/TracingSummit2017>

Userspace callstack view

- Build binary using `-finstrument-functions`
- Enable userspace tracing with LTTng (see GitHub link)
- `LD_PRELOAD="liblttng-ust-cyg-profile-fast.so" ./my_binary`






Demo

Future Work



- Currently at v2.0.0-pre4
 - APIs are not frozen yet
 - Works on all supported platforms
- Targetting the first Release Candidate for November
 - Optimizations which may affect the APIs
 - Documentation
- Support for CTF 2 (v2.1, if the spec is finalized soon)
- Future components
 - Filtering (v2.1)
 - State tracker, Period/span tracking, ideas?

Questions ?

Babeltrace

-  diamon.org/babeltrace
-  github.com/efficios/babeltrace
-  lttng-dev@lists.lttng.org

*Effici*OS

-  www.efficios.com
-  [@efficios](https://twitter.com/efficios)