Cross-Layer Telemetry Support in Linux Kernel

https://github.com/Advanced-Observability/cross-layer-telemetry

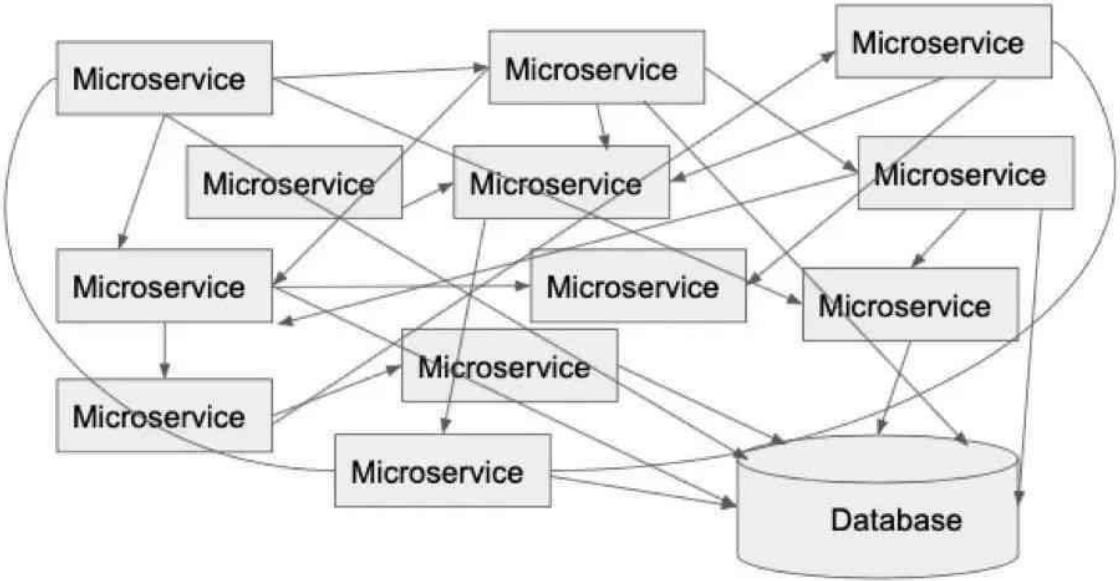
Justin Iurman, Benoit Donnet
<justin.iurman@uliege.be>

Netdev 0x16
October 28, 2022, Lisbon (Portugal)

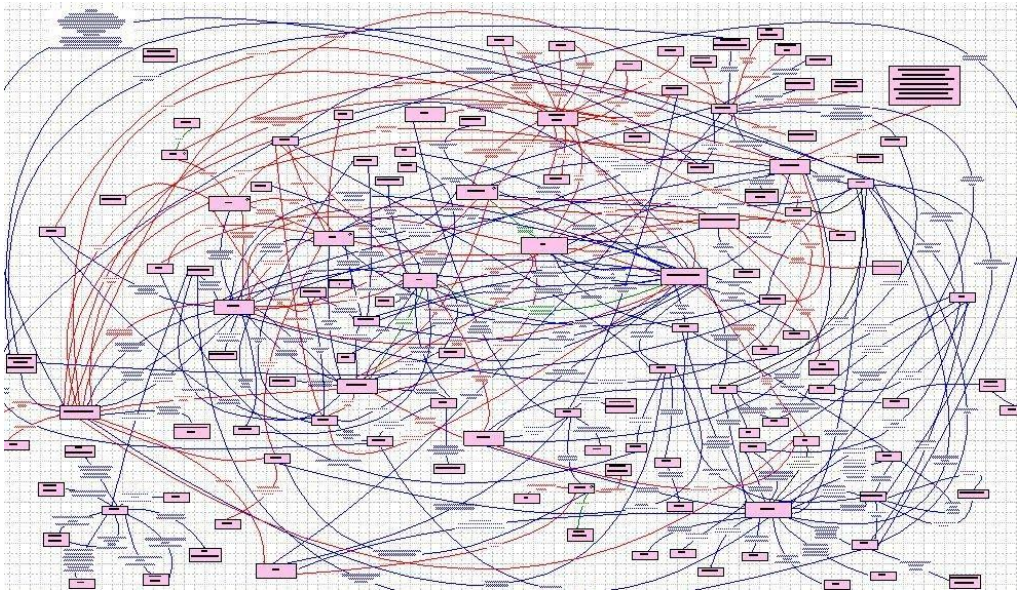
From Monolithic to Microservice architecture

+ flexibility
+ high reliability
+ independently deployable

- debugging challenges



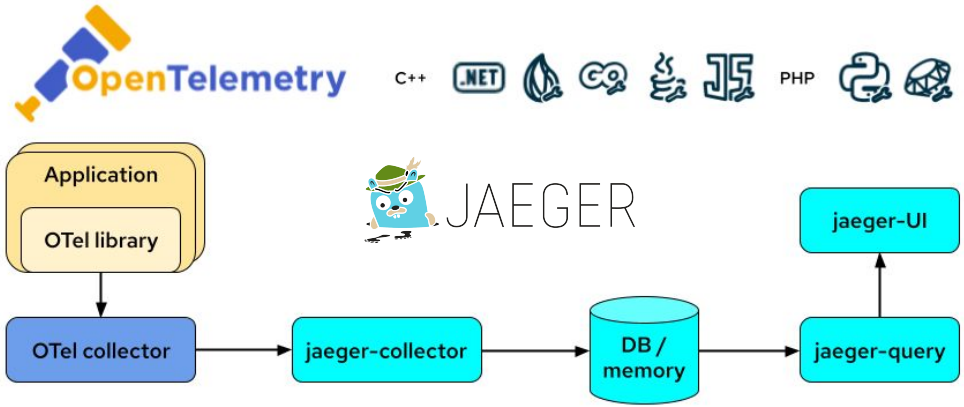
How about that?



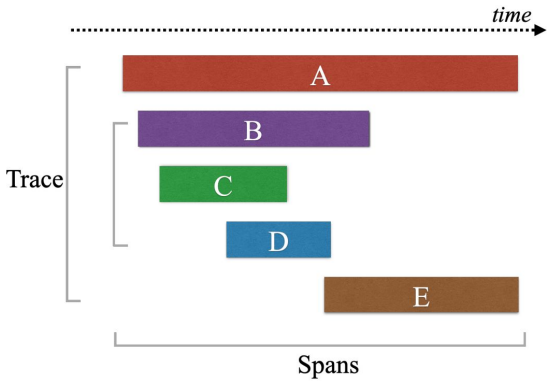
Application Performance Management (APM)

Solution:

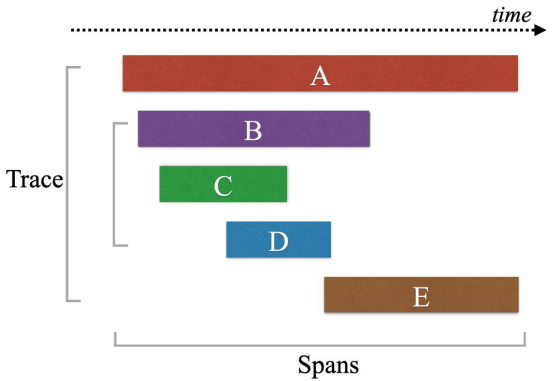
- Distributed tracing (e.g., OpenTelemetry)
- Very useful for (spaghetti-but-not-only) microservices



Tracing world – traces and spans



Tracing world – traces and spans




```python
@app.route('/api/generate_pdf', methods=['GET'])
def generate_pdf():
    with tracer.start_as_current_span('generate_pdf') as parent:
        with tracer.start_as_current_span('check_permissions') as child:
            check_permissions()
        with tracer.start_as_current_span('get_pdf') as child:
            get_pdf()
```

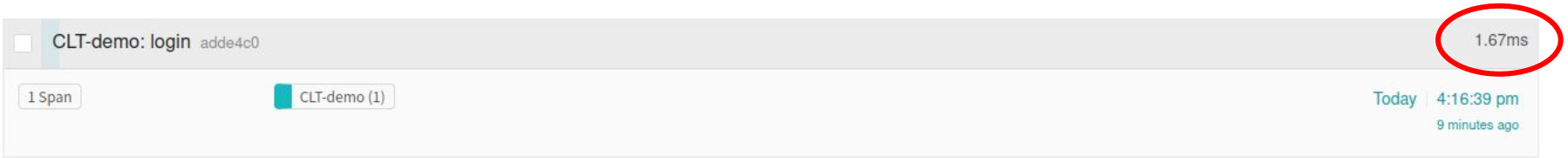
Wait a minute…

What if the monitored block of code

Monitoring an HTTPS request

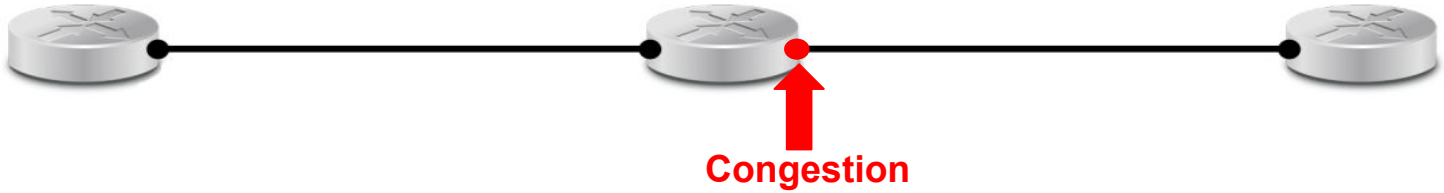
Example: login API

```
@app.route('/api/login', methods=['POST'])
def login():
    with tracer.start_as_current_span('login') as span:
        https.request('POST', '/login', body, headers)
        resp = https.getresponse().read()
```



Monitoring an HTTPS request

Let’s simulate a delay somewhere on the path…



Is it the app, the server, the DB, a network issue, …?

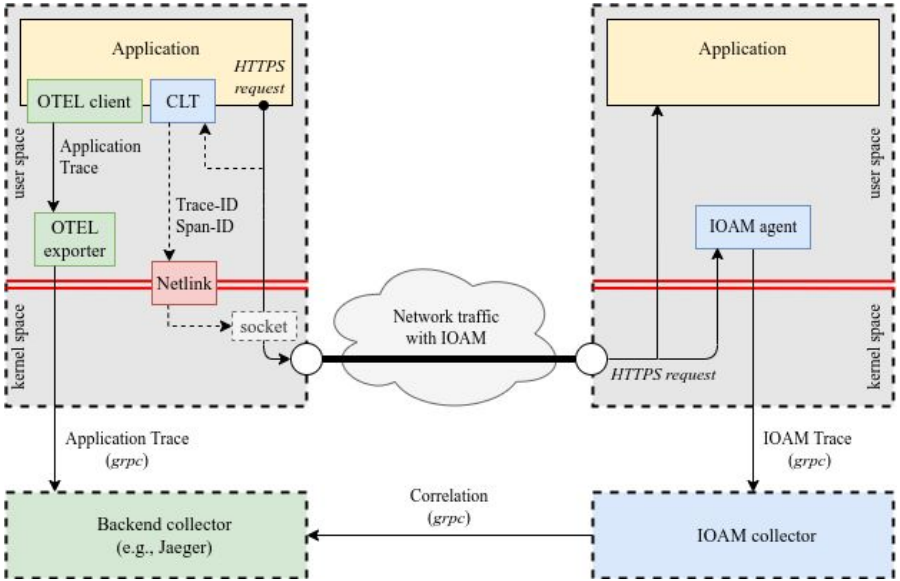
⇒ All we know is that it’s slow… ***why*** ?!

Cross-Layer Telemetry

“*Cross-Layer Telemetry (CLT) makes the entire network stack (L2→L7) visible to monitoring tools, instead of the classic application level visibility*”.

Cross-Layer Telemetry

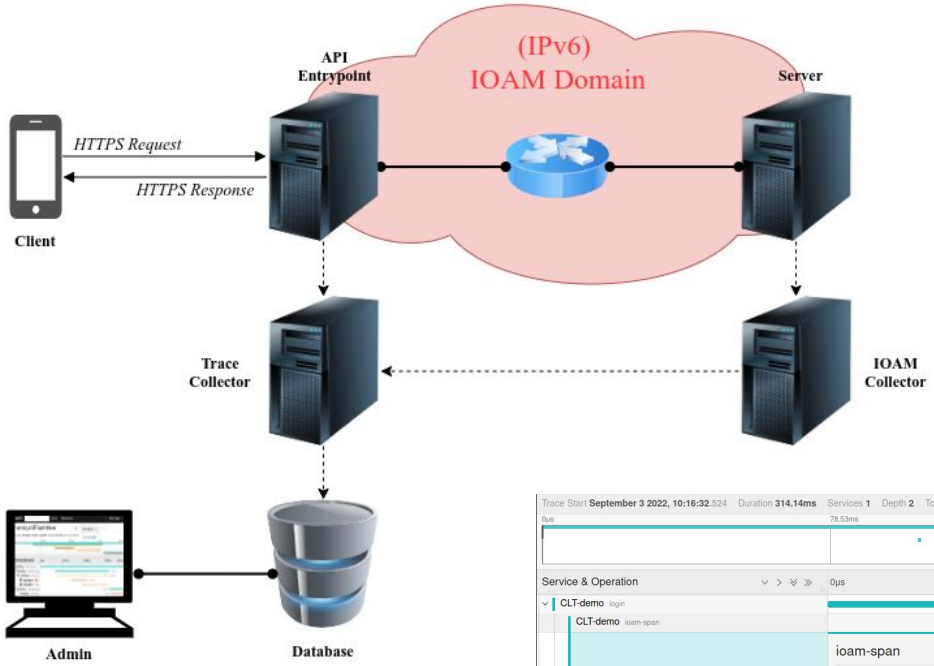
"*Cross-Layer Telemetry (CLT) makes the entire network stack (L2→L7) visible to monitoring tools, instead of the classic application level visibility*".

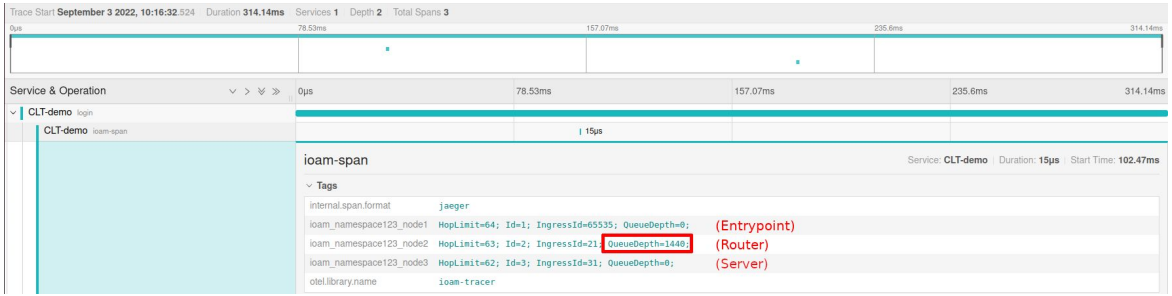


Patch attempt

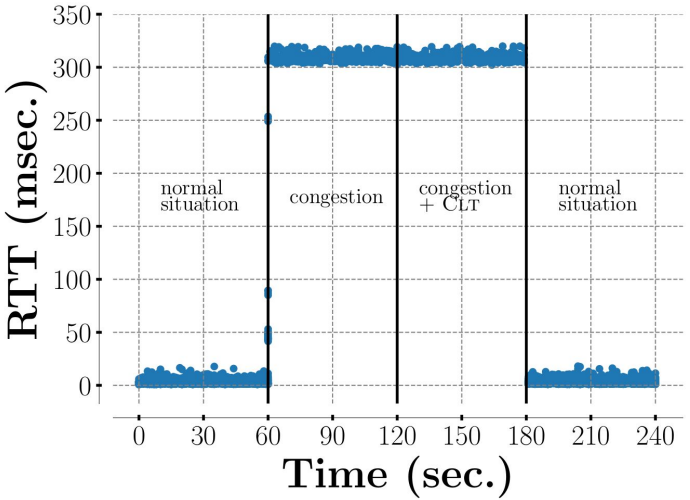
- Quite straightforward (see https://github.com/Advanced-Observability/cross-layer-telemetry/blob/main/CLT.patch)
- Add 2 fields (128 and 64 bits for the trace and span IDs) to:
  - struct sock
  - struct sk_buff
- Add a netlink call to pass IDs from userspace (copy IDs to the corresponding socket)
- Copy IDs from socket to skb in:
  - tcp_sendmsg_locked
  - (udpv6_sendmsg)
- Insert IDs within IOAM (tmp solution)

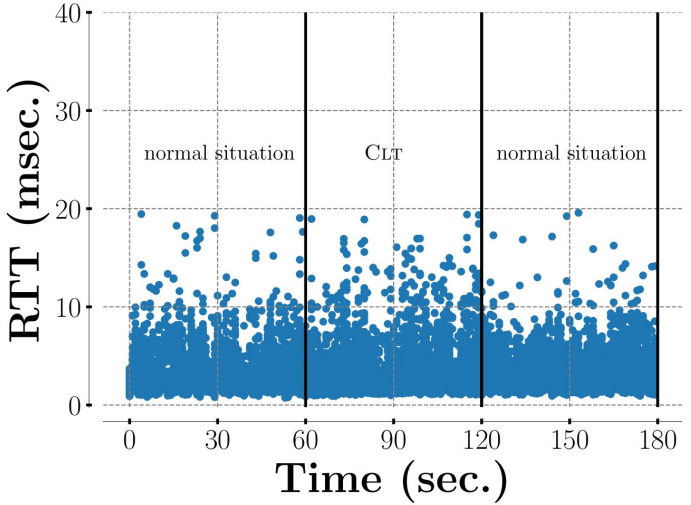
OpenTelemetry/Jaeger on steroids





Performances





IETF work

“Carrying a Generic Identifier in IPv6 packets” [I-D.draft-iurman-6man-generic-id]
https://datatracker.ietf.org/doc/draft-iurman-6man-generic-id

→ potential solution to CLT standardization instead of using IOAM to carry IDs

Conclusion

- Useful
- Any interest to have support in the kernel?
- Thoughts on patch design?

Thank you

https://github.com/Advanced-Observability/cross-layer-telemetry