Overview and Backwards Compatibility of OpenTracing and OpenCensus

Steve Flanders (Omnition)

CloudNativeCon China, Shanghai 2019
Steve Flanders
Head of Product and Experience, Omnition
@smflanders | https://sflanders.net
Background: OpenTracing and OpenCensus
## Cloud-native telemetry

### Telemetry "layers"

<table>
<thead>
<tr>
<th>Instrumentation APIs</th>
<th>Tracing</th>
<th>Metrics</th>
<th>Logs, etc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><code>foreach(language)</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Canonical implementations</th>
<th>Tracing</th>
<th>Metrics</th>
<th>Logs, etc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><code>foreach(language)</code></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data infrastructure</th>
<th>Tracing</th>
<th>Metrics</th>
<th>Logs, etc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><code>collectors, sidecars, etc</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interop formats</th>
<th>Tracing</th>
<th>Metrics</th>
<th>Logs, etc</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><code>w3c trace-context, wire formats for trace data, metrics, logs, etc</code></td>
<td></td>
</tr>
</tbody>
</table>
Overlapping but non-identical

**OPEN TRACING**

- One “vertical” (Tracing)
- One “layer” (API)
- “Looser” coupling (small scope)
- Lots of languages (~12)
- Broad adoption
- (FYI: Already part of CNCF)

**OpenCensus**

- Many “verticals” (Tracing, Metrics)
- Many “layers” (API, impl, infra)
- “Tighter” coupling (framework-y)
- Many languages (5 in beta)
- Broad adoption

(FYI: Already part of CNCF)
<table>
<thead>
<tr>
<th>OPEN TRACING</th>
<th>OpenCensus</th>
</tr>
</thead>
<tbody>
<tr>
<td>- One “vertical” (Tracing) Users want only one dependency</td>
<td>- Many “verticals” (Tracing, Metrics)</td>
</tr>
<tr>
<td>- One “layer” (API)</td>
<td>- Many “layers” (API, impl, infra)</td>
</tr>
<tr>
<td>- “Looser” coupling (small scope)</td>
<td>- “Tighter” coupling (framework-y) Users and vendors want flexibility</td>
</tr>
<tr>
<td>- Lots of languages (~12)</td>
<td>- Many languages (5 in beta)</td>
</tr>
<tr>
<td>- Broad adoption</td>
<td>- Broad adoption</td>
</tr>
<tr>
<td>- (FYI: Already part of CNCF)</td>
<td></td>
</tr>
</tbody>
</table>

Overlapping but non-identical, and some issues
Both widely adopted

**OPEN TRACING**
- 370 contributors
- 5360 stars on GitHub
- 100s of supported integrations with OSS libraries and frameworks

**OpenCensus**
- 390 contributors
- 3392 stars on GitHub
- Backed by Google, Microsoft, Omnition, Postmates, Dynatrace, Shopify (soon)
So both projects are well-adopted! Great! … kinda.
Choice: not always a good thing

No “clear winner” between OpenTracing and OpenCensus
- “Lots of integration” vs “canonical implementation”
- Both projects have “escape velocity”

Ecosystem confused and visibly held back
- Both end-users and vendors
- E.g., this Hadoop ecosystem ticket where the choice was irresolvable

… and all of this made worse by general “negativity bias” on twitter/etc
OpenTelemetry: **the next major version** of *both* OpenTracing and OpenCensus
OpenTelemetry Overview
Cloud-native telemetry

Telemetry “verticals”

<table>
<thead>
<tr>
<th>Telemetry “layers”</th>
<th>Tracing</th>
<th>Metrics</th>
<th>Logs, etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instrumentation APIs</td>
<td>foreach(language)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Canonical implementations</td>
<td>foreach(language)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data infrastructure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interop formats</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OpenTelemetry
Instrumentation APIs

- Context propagation, tracing, metrics and correlation between them. Eventually logs
- Compatibility with existing OT/OC via bridge
- Will rely on and broaden the instrumentation base of OpenTracing/OpenCensus
Canonical implementations

- One “reference implementation” for each language
- W3C trace-context based context propagation
- Open wire format for tracing, metrics, logs
Data infrastructure (coming)

- Support for all popular OSS and commercial backends
- Application and infrastructure metric collection
- For standalone applications (e.g. Envoy or HBase) the backend can be configured in the agent without recompiling
- Tail-based sampling
OpenTelemetry Goals
How standards proliferate:
(See: A/C chargers, character encodings, instant messaging, etc.)

**Situation:**
There are 14 competing standards.

14?! Ridiculous!
We need to develop one universal standard that covers everyone's use cases.

Yeah!

**Soon:**

**Situation:**
There are 15 competing standards.
OpenTelemetry Goals: 2019

- Backwards-compatibility
- Backwards-compatibility
- Backwards-compatibility
- **Sept 2019:** “Time-to-parity” in major langs
- **Nov 2019:** OpenTracing and OpenCensus officially sunsetted (i.e., read-only)
  - Two-year compatibility guarantee
OpenTelemetry Goals: Long-term

- **One** project (not two, and def not three!)
- Broad surface area (tracing, metrics, APIs, reference impls, sidecars, data formats)
- Loose coupling (only take what you need)
- Open governance with representation from many orgs
OpenTelemetry Governance
OpenTelemetry community membership

A SIG for the cross-language specification

SIGs for each language

Three levels of community membership:

1. **Member**: contributor, reviewer
2. **Approver**: experienced reviewer and approver
3. **Maintainer**: set direction and priorities
OpenTelemetry governance

TL;DR edition:

- Using CNCF Code of Conduct
- Intended to represent many organizations / companies
- Elections based on k8s: 9 seats with limits on overrepresentation
  - Active code contributors get to vote
- Maximum term limits
Backwards Compatibility: OpenTracing
Generally, “highly non-abstract” (i.e., in Java it’s a `final` class)
- TraceIDs and SpanIDs much more prescriptive in OpenTelemetry
  - based on w3c standard
- OpenTracing Baggage via w3c’s “correlation-context” concept
- In-process context propagation in its own package
- Introduces a more formal notion of “sampling” (actual methods, not just a standard string key like OpenTracing)
Changes: Span

- A few terminology adjustments, but mostly 1:1 mappings:
  - OpenTracing “tags” become OpenTelemetry “attributes”
  - OpenTracing “logs” become OpenTelemetry “events”
- OpenTracing “References” become OpenTelemetry “parent” Spans that may have varying “kinds” (e.g., Server, Client, Producer, Consumer, Internal)
- OpenTelemetry includes an OpenCensus-like “linked Span” concept: other elements in a batch, etc, etc
- User-provided timestamps for Span start/end not part of OpenTel’s API, though present in wire formats. Can be added if.f. needed.
Changes: Tracer

- Both provide access to the current active Span
- OpenTelemetry introduces the concept of a “Resource” that can attach data to every Span created by the Tracer
- OpenTelemetry will probably not support Tracer.close(), though it probably will support Tracer.flush()
Backwards Compatibility: OpenCensus
OpenTelemetry vs OpenCensus

- Clear separation between the instrumentation API specification and the implementation (alternative implementations can be used)
- A few small changes to the instrumentation API
- OpenTelemetry inherits OpenCensus client library implementations with modifications to adhere to the new API spec
- OpenTelemetry inherits the collection infrastructure from OpenCensus
Transition plan for OpenCensus Service

Receivers ->

service
service
service
service
service
service

Exporters ->

jaeger
zipkin
prometheus
opencensus

oc agent / oc collector
Transition plan for OpenCensus Service

![Diagram showing the transition plan with services, receivers, and exporters connected to Jaeger, Zipkin, Prometheus, and OpenCensus collectors.]
Transition plan for OpenCensus Service

Receivers ->

Exporters ->

service

jaeger
zipkin
prometheus
opencensus
opentelemetry

opentel agent / opentel collector

service

service

service

service

service
Transition plan for OpenCensus Service

Receivers ->

- jaeger
- zipkin
- prometheus
- opencensus
- opentelemetry

Exporters ->

- jaeger
- zipkin
- prometheus
- opencensus
- opentelemetry
What’s Next

- First version of OpenTelemetry by end of 2019, support for 5 languages
- Bridge between OpenTelemetry and OpenTracing/OpenCensus
- OpenTracing and OpenCensus sunset (read-only) by end of 2019
- OpenTracing and OpenCensus backwards compatible through 2021
- OpenTracing terminology, context propagation, and sampling changes
- OpenCensus clear separation from API and implementation
OpenTelemetry: Q & A