Overview

- About Bloomberg: Who are you?
- Recap of Distributed Tracing: What’s this new craze about?
- What tracing systems look like: I only read the abstract...
- Tracing at Bloomberg: What’s your spin on it?
- What’s “Real-time Data Streaming”?: Why are you so special?
- What challenges we faced?: Was it really that hard?
- What’s next?: Where are you going with this?
About Bloomberg
What is the Bloomberg Terminal?
About Bloomberg

Bloomberg is a **technology company** that aims to facilitate financial decision making by providing transparency to global markets.

- Around **20,000** employees globally with **5,500+** engineers
- One of the largest private networks in the world
- **100 billion** market data “ticks” processed daily
- Focus on innovation and **long-term** growth
- **Philanthropy** at the heart of our organisation

TechAtBloomberg.com
Recap of Distributed Tracing
Example - A Search Engine

Input Query
Main Search
News Search
Classification
Entity Lookup
Media Lookup

©2018 Google LLC, used with permission. Google and the Google logo are registered trademarks of Google LLC.
Trace Concepts - Span Model

**Trace**
The journey of an input/event

**Span**
The work done in some component along the trace

**Carrier**
A protocol (such as RPC/Messaging Layer) which communicates trace-aware messages
What tracing systems look like
Tracing Architecture (One Interpretation)

Node

RPC Client Library
Tracer
SDK
App
Tracing Agent

Node

RPC Server Framework
Tracer
SDK
Service
Tracing Agent

Tracing Backend
Distribution
Analytics
Alerting
Monitoring
Storage

Tracing Tools

ZIPKIN
JAEGER

TechAtBloomberg.com
© 2019 Bloomberg Finance L.P. All rights reserved.

Bloomberg Engineering
Using Trace

- Triage/Debugging
- Analysis
- Visualization
- Metrics?
- ...

TechAtBloomberg.com
Tracing at Bloomberg
Tracing Model & Implementation

● Mostly following OpenTracing specification
● Custom library implementations in targeted languages, primarily C++ due to widespread usage
● Own agents and distribution
  ○ Built on-top of existing telemetry infrastructure (metrics, logs)
● Integration with open source tooling
  ○ JaegerUI reads from our back-end
Our Journey

- Started with learning from Dapper
- Tried targeting isolated sub-systems
- Need top-down complete coverage
- Minimize work for app teams
- Build tracing into app frameworks
- Add tracing support for middleware

What’s Next for Tracing?

- Instrument more middleware
- More app frameworks/languages
- Improve our storage
- More flexible head-based sampling
- Tail-based sampling and analysis
- Increase adoption by training
Real-time Data Stream Applications
A Trading System

I’d like to buy some stocks!
System Characteristics

Real-time Streaming
Most traffic uses event-based streaming protocols (usually no polling)

Session Based Subscriptions
Stateful sessions/active connections are common, initial load, caching

Fan-Out/Fan-in Distribution Patterns
For complex intersecting dataflows

Flow Control
Commonly utilise: throttling, batching, chunking, conflation
How we traced it?

- We used the **Span Model**
- Embed tracing into our communication libraries
  - Implement **Carrier**
  - Create **Spans**
- Create **Traces** in application code
- Manually instrument **untraced middleware**
- Context propagation in libraries and application code
- Relationships are usually: **FollowsFrom**

TechAtBloomberg.com
What does it look like?

[Image of a Jaeger UI showing a trace of ApplicationServer.tsx handleRequest 97250a3 with a Request - Response and Message/Streaming (async) highlighted]
The interesting bit

Async event handling
Fan-out message flow
Intra-process latency (nodes)
Inter-process latency (edge)
Sub-system latency
What we do with it?

- Issue analysis (debugging, triaging)
- Metrics extraction (and events)
- Basis for (sub)system Service Level Indicators
- Advanced analysis
  - Identifying dependencies
  - Interactive Jupyter notebook powered workflows
  - Join with other telemetry data (logs)
Challenges
Volume!

- A span is large! (let's say 1KB)
- We generate 500 Million spans a day
- Storage for 30 days 15 Billion spans

Using cloud databases you would be looking at approx $20,000/month

- A lot of data to analyse
- Why not sample?

TechAtBloomberg.com
#1 Message Fan-Out (broadcast)

- Inefficient Distribution
- Late stage filtering (up to 80% discard)
- Redundancy (hot/warm replicas)
- Results in **Noisy** traces!

Solution: Cancel the Span Collection

```python
def handle_message(message):
    span = scope_manager.get_current_span()
    if not is_relevant(message):
        span.cancel()
    return
# ...
```

TechAtBloomberg.com
#2 Splitting Messages

- Multi-part messages
- Can take different paths
- Large messages (initial paint)
- Handled independently (timing)

Solution: Create new spans

“Dispatch” Spans
#3 Message Conflation

- Multiple upstream sources
- High rate of messages
- Often only **last value** relevant
- Data is often flattened/conflated
- Published on interval “flush()”

**Solution: Use “conflation” spans**

- The “distant relationship”
- Tooling support? **Trace View**
#4 Increasing Granularity

- Spans are **Expensive**
- Tags/Attributes are not

## Solution: Span-like Tag Semantics

- TimeSpans - Linear Span
- CheckPoints - Time Reading

```python
from traceutil import timespan, checkpoint

def handle_message(message):
    span = scope_manager.get_current_span()
    with timespan("DoingSomething", span):
        do_something()
        do_something_else()
        checkpoint("DoneSomethingElse", span)
        # ...
```

```json
{
    "operationName" : "publishMessage",
    "tags" : [
        { "key" : "convertMessage_finish", "value" : "20AUG2019_09:30:08.710575+0000" },
        { "key" : "convertMessage_start", "value" : "20AUG2019_09:30:08.710537+0000" },
        { "key" : "publishMessageInterleaver_checkpoint", "value" : "20AUG2019_09:30:08.710400+0000" }
    ]
}
```
#5 - Sampling

- **Head-based** (trace creation time)
- **Unitary** (specific components)
- **Tail-based** (once trace is known)
What’s Next?
What’s Next?

● Trace More!

● Tracing the Client-Side (heavily async)

● Near real-time trace processing
  ○ Derive Metrics (and events)
  ○ Alerting
We are hiring!

https://www.bloomberg.com/careers

Thank you for listening!

Questions?

TechAtBloomberg.com