Tracing: Fast & Slow
Digging into and improving your web service’s performance

Offline viewers: full write up @ rogue.ly/tracing
$ whoami
agenda
agenda

- Overview and problem space
agenda

- Overview and problem space
- Approaches to tracing
agenda

• Overview and problem space
• Approaches to tracing
• Tracing at scale
agenda

- Overview and problem space
- Approaches to tracing
- Tracing at scale
- Diagnosing performance issues
agenda

- Overview and problem space
- Approaches to tracing
- Tracing at scale
- Diagnosing performance issues
- Tracing services & systems
Tracing Overview
machine-centric

- Focus on a single machine
machine-centric

- Focus on a single machine
- No view into a service’s dependencies
workflow-centric

- Understand causal relationships
workflow-centric

- Understand causal relationships
- End-to-end tracing
why trace?
why trace?

- Performance analysis
why trace?

- Performance analysis
- Anomaly detection
why trace?

- Performance analysis
- Anomaly detection
- Profiling
why trace?

- Performance analysis
- Anomaly detection
- Profiling
- Resource attribution
why trace?

- Performance analysis
- Anomaly detection
- Profiling
- Resource attribution
- Workload modeling
Tracing Approaches
manual
```python
def request_id(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        req_id = request.headers.get(
            "X-Request-Id", uuid.uuid4())
        return f(req_id, *args, **kwargs)
    return decorated

@app.route("/")
@request_id
def list_services(req_id):
    # log w/ ID for wherever you want to trace
    # app logic
```
upstream appserver {
    10.0.0.0:80;
}

server {
    listen 80;
    # Return to client
    add_header X-Request-ID $request_id;
    location / {
        proxy_pass http://appserver;
        # Pass to app server
        proxy_set_header X-Request-ID $request_id;
    }
}
log_format trace '$remote_addr ... $request_id';

server {
    listen 80;
    add_header X-Request-ID $request_id;
    location / {
        proxy_pass http://app_server;
        proxy_set_header X-Request-ID $request_id;
        # Log $request_id
        access_log /var/log/nginx/access_trace.log trace;
    }
}

blackbox
metadata propagation
Tracing at Scale
four things to think about
four things to think about

- What relationships to track
four things to think about

- What relationships to track
- How to track them
four things to think about

• What relationships to track
• How to track them
• Which sampling approach to take
four things to think about

• What relationships to track
• How to track them
• Which sampling approach to take
• How to visualize
what to track
Request One

10μs → cache write
10μs → insert block
10μs → write reply

1 min → evict
10μs → disk write
10μs → block persisted

Request Two

10μs → cache write
10μs → insert block
10μs → write reply

30μs → block persisted

Submitter Flow PoV
Request One

- write
- cache write
- insert block
- write reply

Request Two

- write
- cache write
- evict
- disk write
- block persisted
- insert block
- write reply

Trigger Flow PoV
how to track
request ID
request ID +
logical clock
request ID + 
logical clock + 
previous trace points
tradeoffs
tradeoffs

- Payload size
tradeoffs

- Payload size
- Explicit relationships
Tradeoffs

- Payload size
- Explicit relationships
- Collate despite lost data
tradeoffs

- Payload size
- Explicit relationships
- Collate despite lost data
- Immediate availability
how to sample
sampling approaches

- Head-based
sampling approaches

• Head-based
• Tail-based
sampling approaches

- Head-based
- Tail-based
- Unitary
what to visualize
request flow graph

A call

B call

C call

D call

E call

C reply

B reply

E reply

D reply

A reply

C reply
context calling tree
keep in mind

- What do I want to know?
keep in mind

• What do I want to know?
• How much can I instrument?
keep in mind

• What do I want to know?
• How much can I instrument?
• How much do I want to know?
suggested for performance
suggested for performance

- Trigger PoV
suggested for performance

- Trigger PoV
- Head-based sampling
suggested for performance

- Trigger PoV
- Head-based sampling
- Flow graphs
Diagnosing
questions to ask

- Batch requests?
questions to ask

- Batch requests?
- Any parallelization opportunities?
questions to ask

- Batch requests?
- Any parallelization opportunities?
- Useful to add/fix caching?
questions to ask

- Batch requests?
- Any parallelization opportunities?
- Useful to add/fix caching?
- Frontend resource loading?
questions to ask

• Batch requests?
• Any parallelization opportunities?
• Useful to add/fix caching?
• Frontend resource loading?
• Chunked or JIT responses?
Frameworks, Systems & Services
Frameworks
OpenTracing
OpenCensus
self-hosted
Zipkin (Twitter)
Zipkin (Twitter)

• Out-of-band reporting to remote collector
Zipkin (Twitter)

- Out-of-band reporting to remote collector
- Report via HTTP, Kafka, and Scribe
Zipkin (Twitter)

- Out-of-band reporting to remote collector
- Report via HTTP, Kafka, and Scribe
- Varying propagation support across different languages
Zipkin (Twitter)

- Out-of-band reporting to remote collector
- Report via HTTP, Kafka, and Scribe
- Varying propagation support across different languages
- Limited web UI
def http_transport(span_data):
    requests.post(
        "http://zipkinserver:9411/api/v1/spans",
        data=span_data,
        headers={"Content-type": "application/x-thrift"})

@app.route("/")
def index():
    with zipkin_span(service_name="myawesomeapp",
                     span_name="index",
                     # need to write own transport func
                     transport_handler=http_transport,
                     port=app_port,
                     # 0-100 percent
                     sample_rate=100):
        # do something
Jaeger (Uber)
Jaeger (Uber)

- Local daemon to collect & report
Jaeger (Uber)

- Local daemon to collect & report
- Storage support for only Cassandra
Jaeger (Uber)

- Local daemon to collect & report
- Storage support for only Cassandra
- Limited Web UI
Jaeger (Uber)

- Local daemon to collect & report
- Storage support for only Cassandra
- Limited Web UI
- Varying language support in client libraries
config = Config(...)  
tracer = config.initialize_tracer()

@app.route("/")
def index():
    with tracer.start_span("ASpan") as span:
        span.log_kv({"event": "the answer to", "life": 42})

    with tracer.start_span("ChildSpan", child_of=span) as cspan:
        cspan.log_kv({"event": "don't forget", "towel": True})
honorable mentions

- AppDash
services
Stackdriver Trace (Google)
Stackdriver Trace (Google)

- OpenCensus with gRPC support
Stackdriver Trace (Google)

- OpenCensus with gRPC support
- Forward traces from Zipkin
Stackdriver Trace (Google)

- OpenCensus with gRPC support
- Forward traces from Zipkin
- Storage limitation of 30 days
Stackdriver Trace (Google)

- OpenCensus with gRPC support
- Forward traces from Zipkin
- Storage limitation of 30 days
- Recreate graphs per time period
X-Ray (AWS)
X-Ray (AWS)

- Supports OpenCensus, not OpenTracing
X-Ray (AWS)

- Supports OpenCensus, not OpenTracing
- Growing SDK support across languages
X-Ray (AWS)

- Supports OpenCensus, not OpenTracing
- Growing SDK support across languages
- Lots of flexibility with configuring sampling
X-Ray (AWS)

- Supports OpenCensus, not OpenTracing
- Growing SDK support across languages
- Lots of flexibility with configuring sampling
- Send metrics from outside AWS environment
X-Ray (AWS)

- Supports OpenCensus, not OpenTracing
- Growing SDK support across languages
- Lots of flexibility with configuring sampling
- Send metrics from outside AWS environment
- Flow graphs with latency, response %, sample %
honorable mentions

- LightStep
- SignalFX
- New Relic
- Datadog
- Azure Monitor
TL;DR
• You need this
tl;dr

- You need this – but it’s hard
tl;dr

- You need this – but it’s hard
- Support is improving
tl;dr

• You need this – but it’s hard
• Support is improving
• One size fits all approaches
tl;dr

• You need this – but it’s hard
• Support is improving
• One size fits all approaches
• It’s in the open
Thanks!

Write up: rogue.ly/tracing

Visit our booth – we’re hiring!

Lynn Root | SRE | @roguelynn