Enhancing Performance Tracing and Debugging in Remote Deployments

September 18, 2023

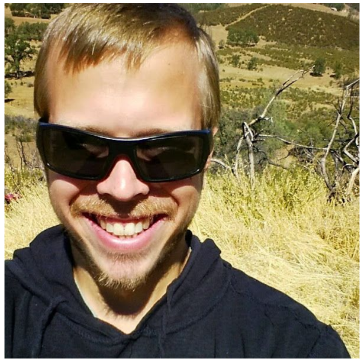
Google Cloud

Contents

Speakers

Alankrit Kharbanda

GCP Data Transfer



AJ Ortega

GCP Data Transfer

Introduction

Intro to tracing

Identifying the problem

- Tracing in On-premises remote deployed containers.
    - Tracing across multiple instances
- Machine not accessible to developer.
- Any changes including setting logstash or alternates is not possible because we deploy a standalone binary.
- Problem is three fold :
    - Generating Traces
    - Moving Traces to the cloud
    - Analyzing Traces
- Get everything together - Metrics , Traces and Logs

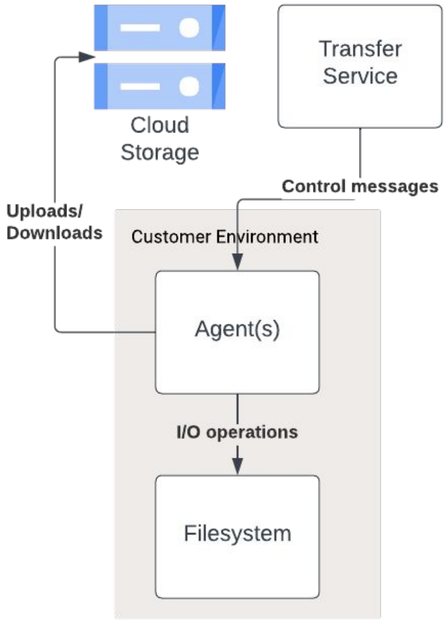
Existing Solutions

Cloud Tracing	• Great for tracing across GCP services and more • Great for latency tracking and traces.
Other cloud tracing	• Powerful distributed tracing, headers in calls to other other cloud services.

- Both these services are great for Traces. Aggregation and analysis is easier with Bigquery using our method.

Solving Tracing in multi-container deployments

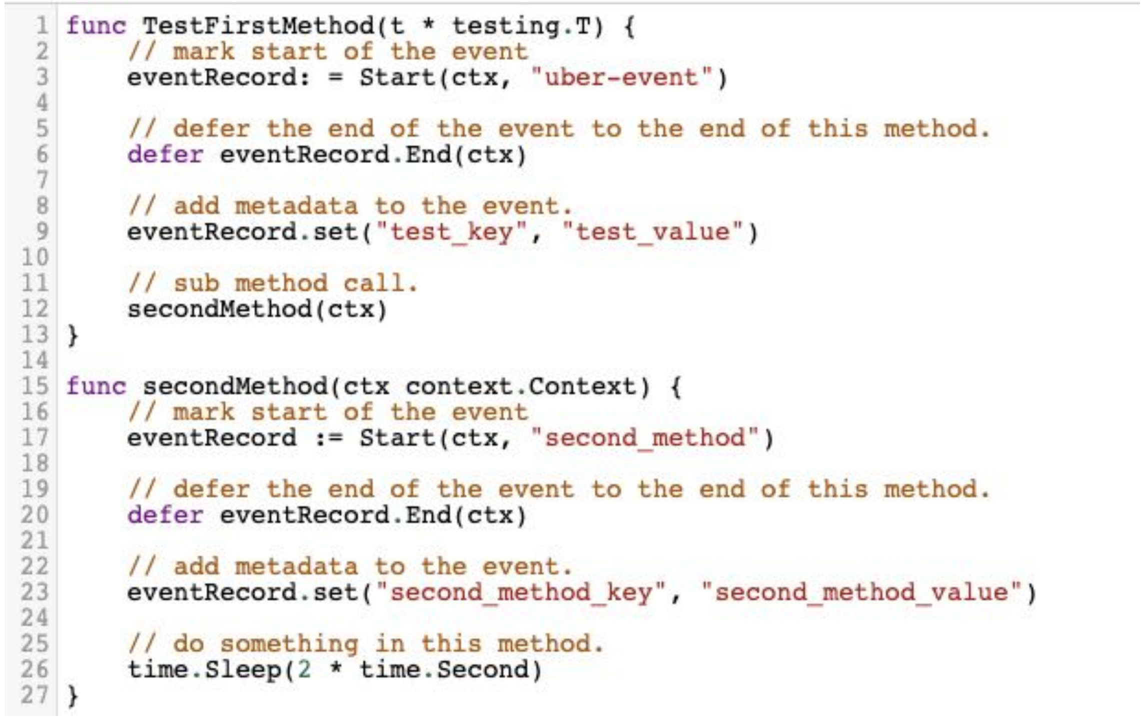
- Trace Beginning and end of methods
- Add metadata as key-value pairs
  - Metadata is tracked for current method/call.
  - Useful to add valuable side information.
  - Metadata can be any scalar type
- Attach ids to Context as "Spans"
  - Spans are tracked on subcalls as well
  - Useful in tracking sub-events to an event



Trace Logs

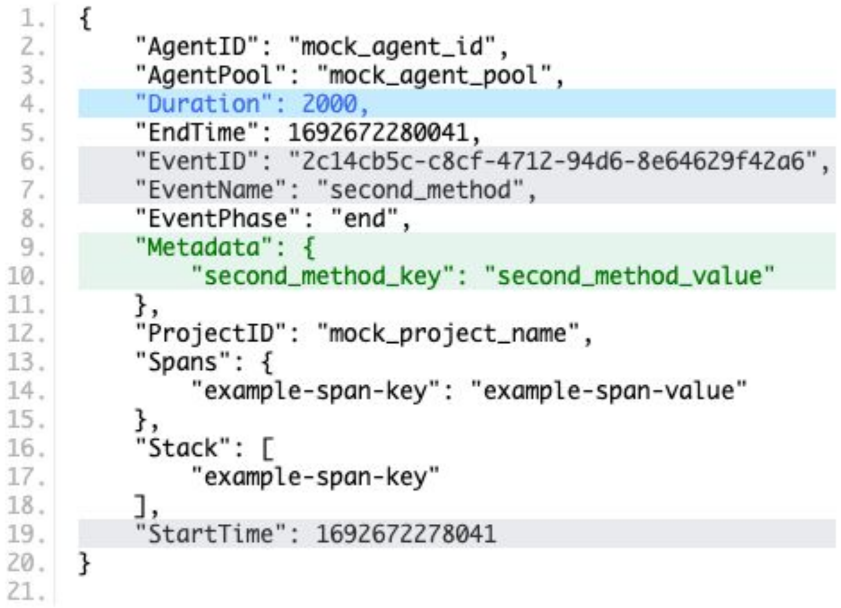
Sample code and Log entries

Code Sample(go)

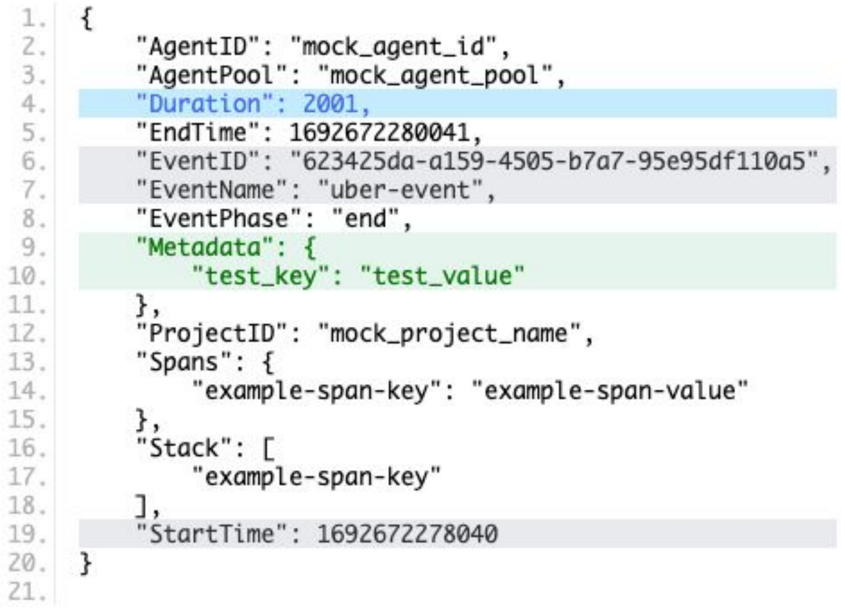


```go
func TestFirstMethod(t * testing.T) {
    // mark start of the event
    eventRecord: = Start(ctx, "uber-event")

    // defer the end of the event to the end of this method.
    defer eventRecord.End(ctx)

    // add metadata to the event.
    eventRecord.set("test_key", "test_value")

    // sub method call.
    secondMethod(ctx)
}

func secondMethod(ctx context.Context) {
    // mark start of the event
    eventRecord := Start(ctx, "second_method")

    // defer the end of the event to the end of this method.
    defer eventRecord.End(ctx)

    // add metadata to the event.
    eventRecord.set("second_method_key", "second_method_value")

    // do something in this method.
    time.Sleep(2 * time.Second)
}
```

Event Sample (ends only)



```
{
    "AgentID": "mock_agent_id",
    "AgentPool": "mock_agent_pool",
    "Duration": 2000,
    "EndTime": 1692672280041,
    "EventID": "2c14cb5c-c8cf-4712-94d6-8e64629f42a6",
    "EventName": "second_method",
    "EventPhase": "end",
    "Metadata": {
        "second_method_key": "second_method_value"
    },
    "ProjectID": "mock_project_name",
    "Spans": {
        "example-span-key": "example-span-value"
    },
    "Stack": [
        "example-span-key"
    ],
    "StartTime": 1692672278041
}
```



```
{
    "AgentID": "mock_agent_id",
    "AgentPool": "mock_agent_pool",
    "Duration": 2001,
    "EndTime": 1692672280041,
    "EventID": "623425da-a159-4505-b7a7-95e95df110a5",
    "EventName": "uber-event",
    "EventPhase": "end",
    "Metadata": {
        "test_key": "test_value"
    },
    "ProjectID": "mock_project_name",
    "Spans": {
        "example-span-key": "example-span-value"
    },
    "Stack": [
        "example-span-key"
    ],
    "StartTime": 1692672278040
}
```

In stream vs out of stream

How to ship logs

Ship logs instream

- Google Cloud Logging
  - Use Cloud Logging SDK
  - Buffer Logs In Memory (Drop excess)
  - Pros :
    - No File Management
    - Easy SDK integration
    - Easy Integration with Bigquery for analysis
  - Cons :
    - Memory limitations by your application
    - Logs maybe dropped

Ship logs out of stream

- Google Cloud Storage
    - Write Trace logs to file
    - Separate threads to compress files
    - Separate threads to send logs to GCS
    - Pros :
        - Way less memory utilization, garbage generation.
        - Choose compression algorithm to optimize speed vs compressed file size.
        - Easy Integration with Bigquery for analysis
    - Cons :
        - Delay in shipping and compressing files -> <5 minutes
        - Manually pull in for analysis.

Sampling - yes or no?

- Optional
- While sampling is great, our usecase was to solve for each trace
- Usecases where tracking all traces is helpful
    - Identifying events which never closed
    - Aggregation
        - Test statistics in aggregate
        - Hotspotting and concurrency trends
        - Pattern analysis for minority anomalies (eg. some small files causing problems vs most files are large).

Analysis

Lets import to Bigquery!

Analyze logs

- Bigquery
- Import logs
    - Cloud Logging -> Direct Import Via Sink
    - GCS Bucket -> Manual import -> Can import gz files.
- Queries
    - Can save project relevant query templates directly
    - Can write compounded queries easily -> Use temp tables if not great at sql to simplify problems
    - Constant analysis, one time analysis.
- Visualize
    - Create sample dashboards/charts. See Demo.

Real life problems we solved

Some problems we solved

- Identify long poles in subprocesses
    - Example - identified an issue in our os.Stat operation
    - OS level issue - would not have detected otherwise.
- Identify Concurrency and Hotspot issue
    - Identified that one of our systems was not honoring thread count limit
    - We were hammering a service we weren’t supposed to.
- Close the loop measurement.
    - Identified that we were not releasing a lock that we were establishing.

Demo !

Google Cloud

Video Demo here

Thank you

Google Cloud