Team 12

Reading: #12 Distributed Tracing
Prototyping: Fog-Alert system

Karl Wolf, Michael Narodovitch, Pavan Chitradurga Thammanna

Prototyping Assignment: Fog-Alert

Prototyping Assignment

Scenario: Managing cold storage houses in remote locations

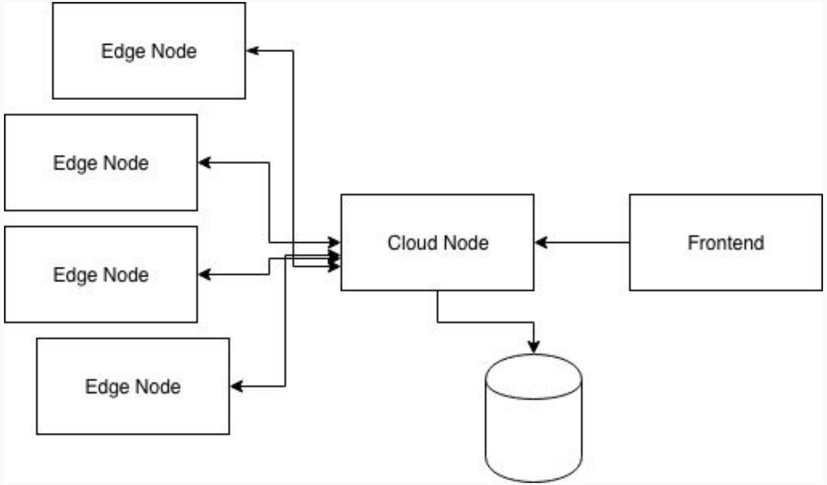
Solution: Edge devices measure temperatures

- Local Alert
- Report to Cloud Backend

Prototyping Assignment

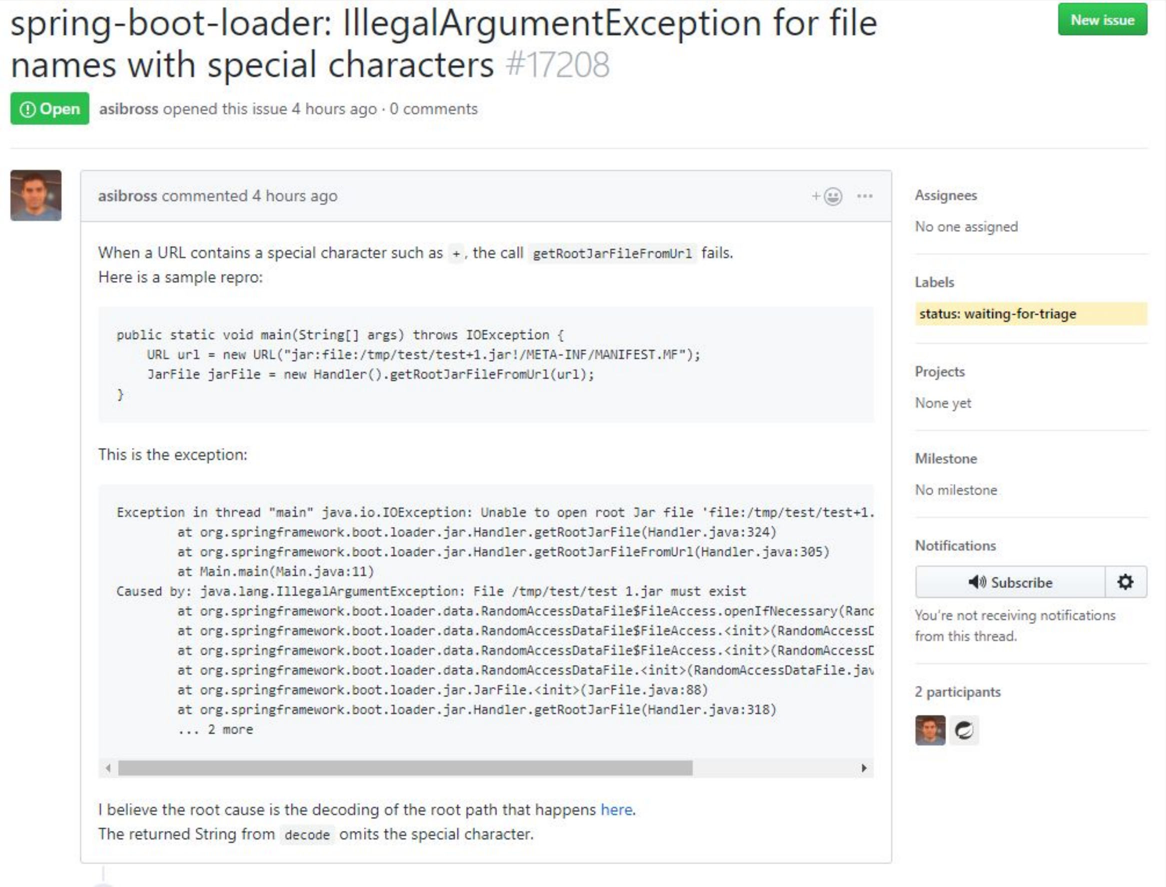
Fog Alert

- Environmental Data using Tinkerforge Sensors
- Sent live to cloud; On error: buffer & resend!
- Write to disk in case of power outage
- Chaos Penguin to make sure buffering works



Reading Assignment: Googles Dapper

Motivation for Tracing



spring-boot-loader: IllegalArgumentException for file names with special characters #17208

asibross opened this issue 4 hours ago · 0 comments

asibross commented 4 hours ago

When a URL contains a special character such as `+`, the call `getRootJarFileFromUrl` fails.
Here is a sample repro:

```
public static void main(String[] args) throws IOException {
    URL url = new URL("jar:file:/tmp/test/test+1.jar!/META-INF/MANIFEST.MF");
    JarFile jarFile = new Handler().getRootJarFileFromUrl(url);
}
```

This is the exception:

```
Exception in thread "main" java.io.IOException: Unable to open root Jar file 'file:/tmp/test/test+1.
        at org.springframework.boot.loader.jar.Handler.getRootJarFile(Handler.java:324)
        at org.springframework.boot.loader.jar.Handler.getRootJarFileFromUrl(Handler.java:305)
        at Main.main(Main.java:11)
Caused by: java.lang.IllegalArgumentException: File /tmp/test/test 1.jar must exist
        at org.springframework.boot.loader.data.RandomAccessDataFile$FileAccess.openIfNecessary(Ranc
        at org.springframework.boot.loader.data.RandomAccessDataFile$FileAccess.<init>(RandomAccessD
        at org.springframework.boot.loader.data.RandomAccessDataFile$FileAccess.<init>(RandomAccessD
        at org.springframework.boot.loader.data.RandomAccessDataFile.<init>(RandomAccessDataFile.jav
        at org.springframework.boot.loader.jar.JarFile.<init>(JarFile.java:88)
        at org.springframework.boot.loader.jar.Handler.getRootJarFile(Handler.java:318)
        ... 2 more
```

I believe the root cause is the decoding of the root path that happens here.
The returned String from `decode` omits the special character.

Assignees
No one assigned

Labels
status: waiting-for-triage

Projects
None yet

Milestone
No milestone

Notifications

You're not receiving notifications from this thread.

2 participants

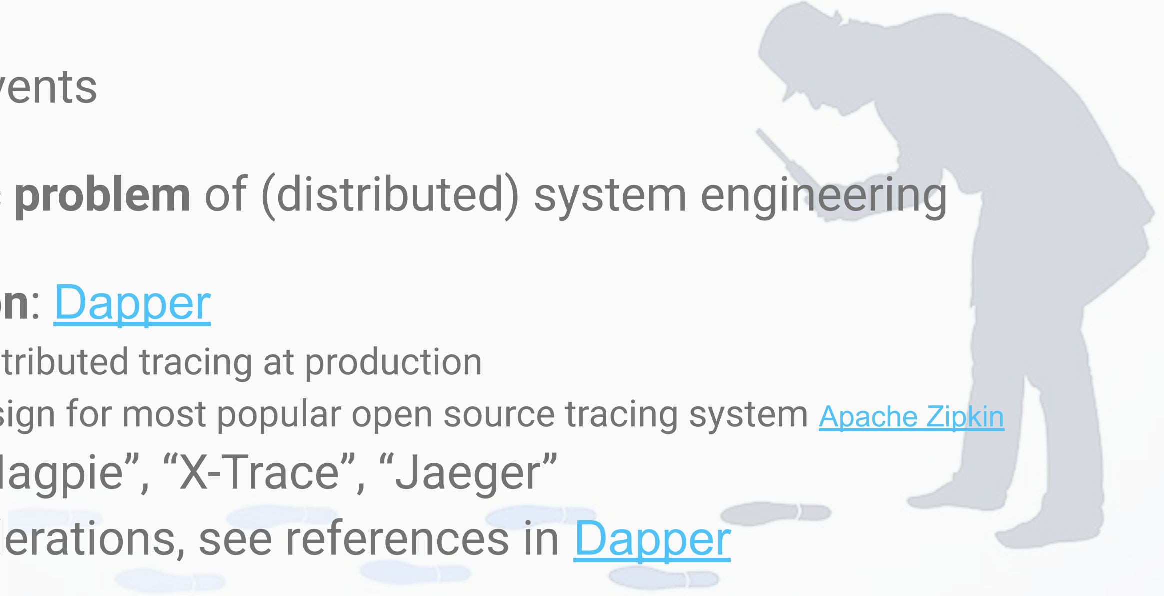
Distributing Tracing - Introduction

Debugging of distributed applications is **hard** - especially when **we do not know**

- which services are involved
- what the services do
- what is the causality of events

Distributed tracing is a **classic problem** of (distributed) system engineering

- Google presents a **solution**: Dapper
  - Google uses Dapper for distributed tracing at production
  - Dapper is the reference design for most popular open source tracing system Apache Zipkin
- Other tracing systems: “Magpie”, “X-Trace”, “Jaeger”
- For general design considerations, see references in Dapper



*https://www.vehicletrackingexperts.co.uk/is-it-illegal-to-track-your-spouses-or-childs-car/

Distributing Tracing - Requirements

Requirements for distributing tracing system

- Ubiquitous deployment
- Continuous monitoring

Design goals for distributing tracing system

- Low overhead
- Scalability
- Low-latency-availability of tracing data
- Application-level transparency

*https://www.vehicletrackingexperts.co.uk/is-it-illegal-to-track-your-spouses-or-childs-car/

Dapper Design - Data Structures

Span

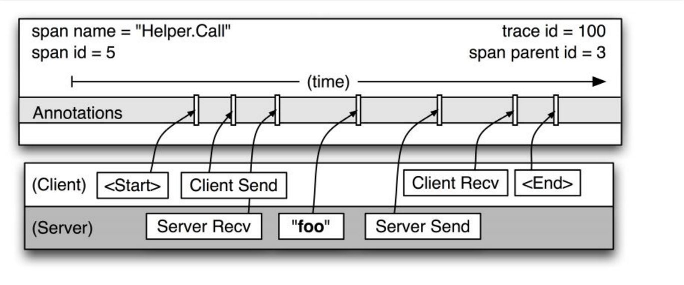
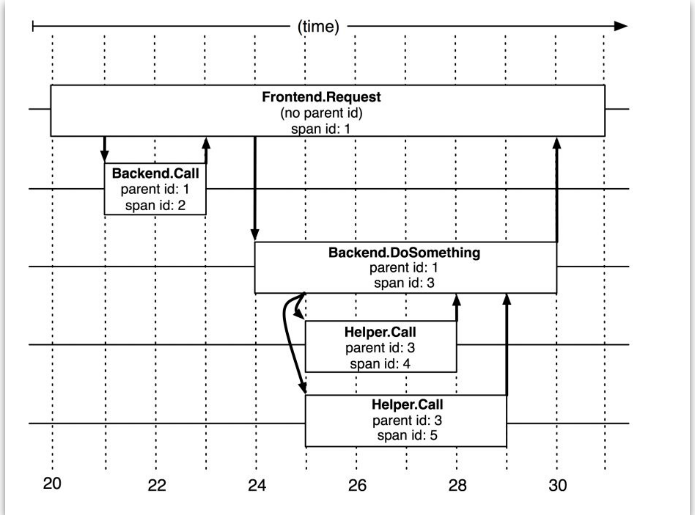


Figure 3: A detailed view of a single span from Figure 2.

Trace Tree



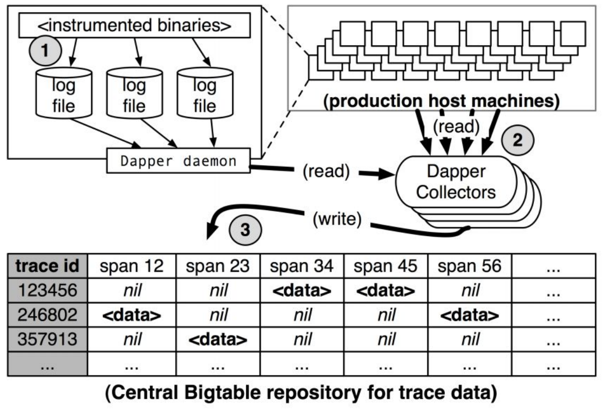
- All servers communicate via GRPC
- Span is generated by GRPC instrumentation
- Annotations are added to the span by the application code (Dapper SDK…)
- Trace tree resembled from `span_id` and `parent_id`
- Globally unique ID for root spans

https://ai.google/research/pubs/pub36356

Dapper Design - Trace Collection pipeline

The trace collection pipeline is characterized by a three stage process:

1. Dapper instrumentation writes to local log file
2. Dapper collectors pull log files from Dapper daemons
3. Dapper collectors push TraceTrees and Spans to BigTable repository



trace id	span 12	span 23	span 34	span 45	span 56	...
123456	*nil*	*nil*	**<data>**	**<data>**	*nil*	...
246802	**<data>**	*nil*	*nil*	*nil*	**<data>**	...
357913	*nil*	**<data>**	*nil*	*nil*	*nil*	...
...	...	...	...	...	...	...

(Central Bigtable repository for trace data)

Figure 5: An overview of the Dapper collection pipeline.

https://ai.google/research/pubs/pub36356

Dapper at Google Sites - Performance at Scale

Design goals for distributing tracing system

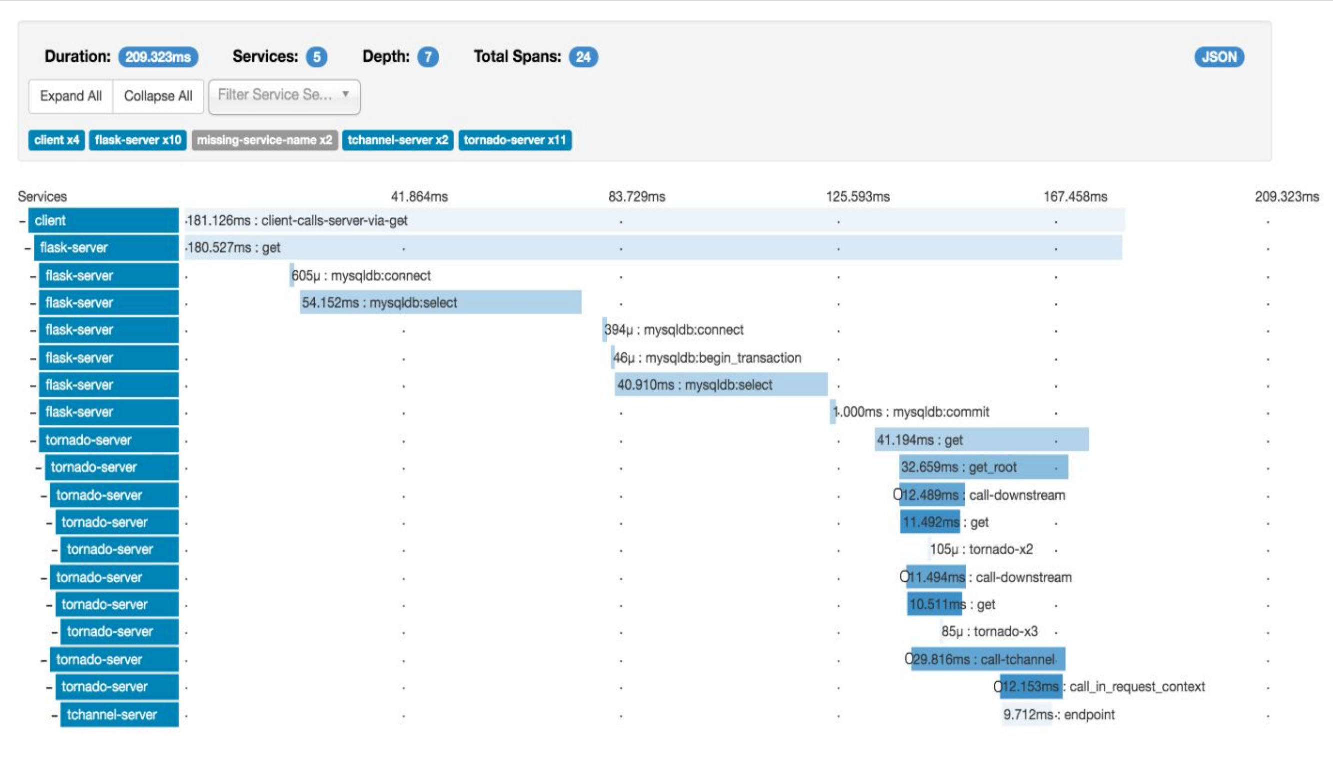
- Low overhead
- Scalability
- Low-latency-availability of tracing data
- Application-level transparency

Reality at Google:

- Footprint of the tracing system
    - < 0.01% Network traffic, <0.3% CPU
    - Memory footprint within the noise of heap fragmentation
- Dapper Daemons deployed on >10.000 server instances
- Typical <1s end-to-end-latency for traces
- Application-level transparency by instrumentation of GRPC libraries

https://www.vehicletrackingexperts.co.uk/is-it-illegal-to-track-your-spouses-or-childs-car/

Apache Zipkin



Duration: 209.323ms Services: 5 Depth: 7 Total Spans: 24

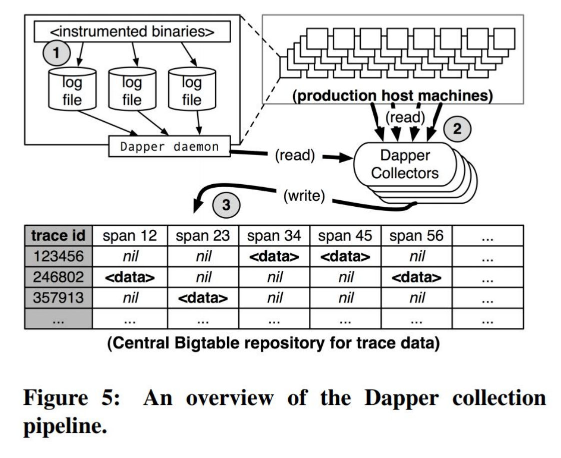
https://zipkin.io/

In Zipkin Repo:

- 2029 commits
- 11,145 stars
- 237 open issues

- Based on Dapper design
- Instrumentation for (almost) every language and any transport

Evolution to the Fog

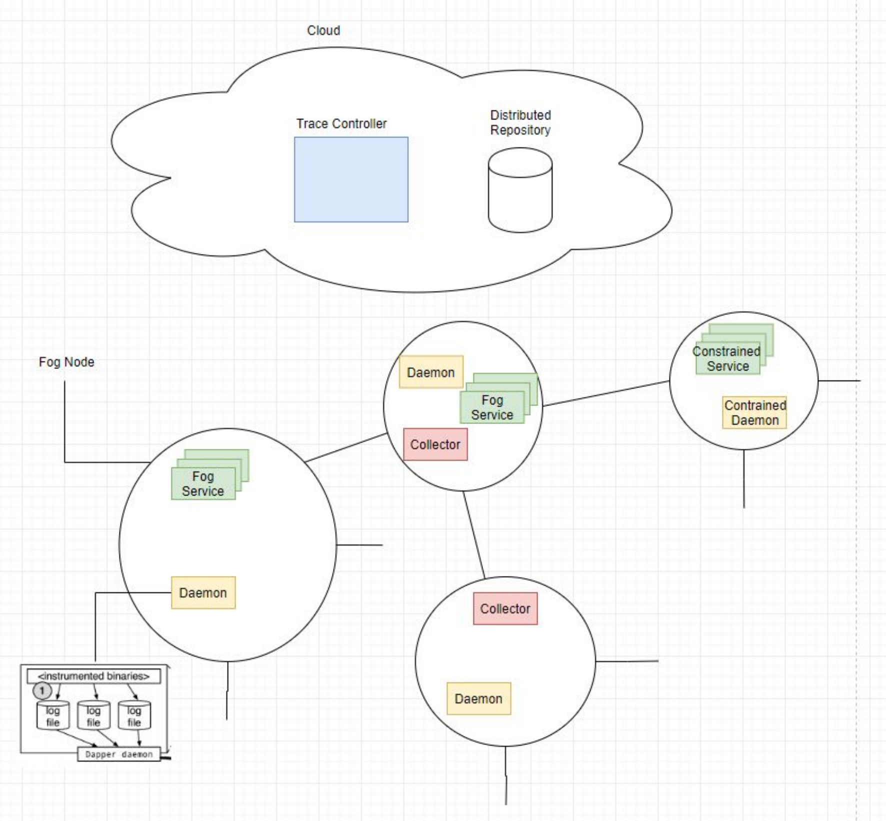


trace id	span 12	span 23	span 34	span 45	span 56	...
123456	*nil*	*nil*	**<data>**	**<data>**	*nil*	...
246802	**<data>**	*nil*	*nil*	*nil*	**<data>**	...
357913	*nil*	**<data>**	*nil*	*nil*	*nil*	...
...	...	...	...	...	...	...

(Central Bigtable repository for trace data)

Figure 5: An overview of the Dapper collection pipeline.

Fog Evolution



- Convergence of collectors and daemons to the same Fog-Node
- Location awareness collectors and daemons
- Traces still pushed to the cloud
- Trace controller for live testing