ZIPKIN TRACING APACHE CASSANDRA

Mick Semb Wever

TWITTER @mck_sw
mick@**thelastpickle.com**

ABOUT THE LAST PICKLE

WORK WITH CLIENTS TO DELIVER AND IMPROVE

APACHE CASSANDRA BASED SOLUTIONS

BASED IN
USA, NEW ZEALAND, AUSTRALIA, FRANCE, LONDON

distributed tracing

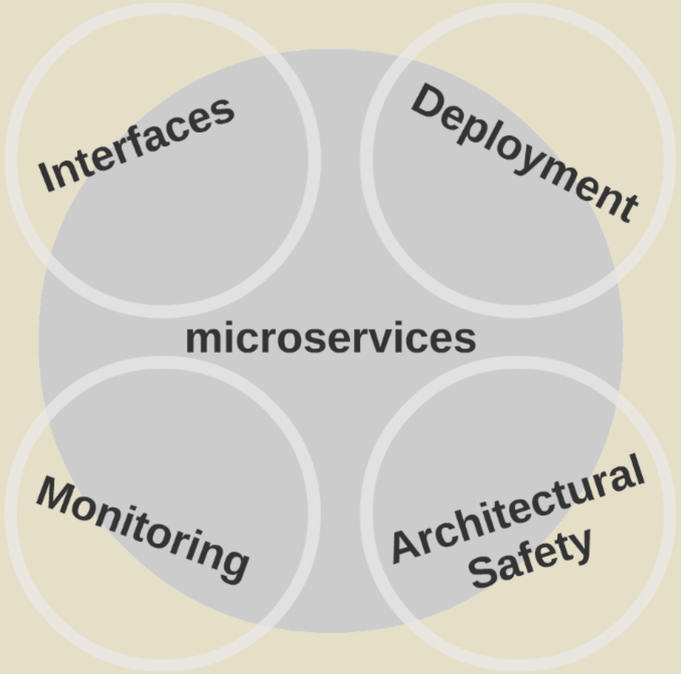
zipkin

cassandra's tracing

zipkin & cassandra

SCALING DATA & PEOPLE

Apache Cassandra
data platform de jure
microservices, base, and lambda architectures



Kibana

Grafana

distributed tracing?

Kibana

Grafana

Zipkin

the missing piece for many is tracing and profiling difficult to reproduce problems

ZIPKIN

an implementation of Google's Dapper paper

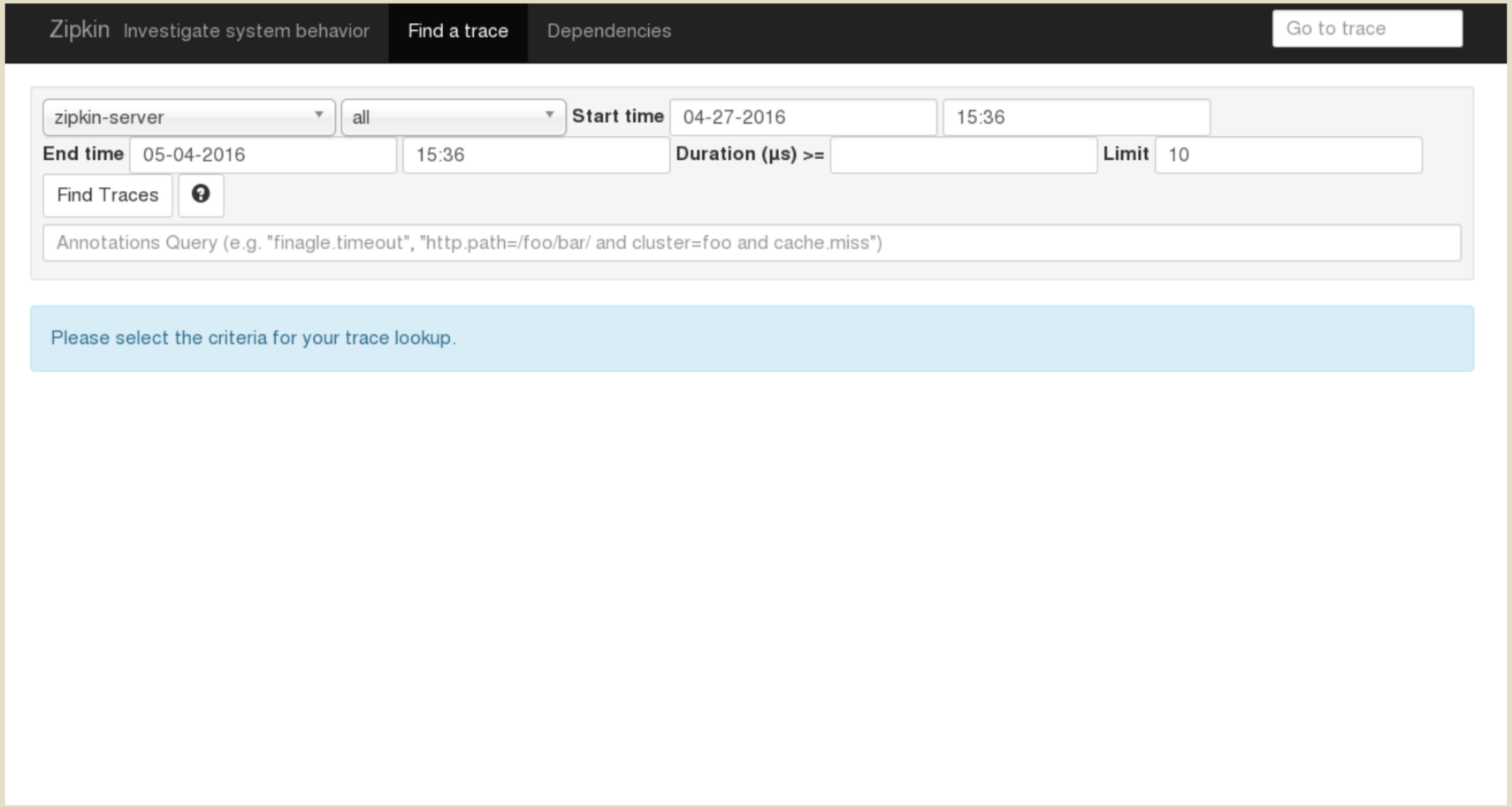
ZIPKIN INSTALL

```
git clone git@github.com:openzipkin/zipkin.git

# doh! it's in the readme

mvn install -DskipTests

STORAGE_TYPE=cassandra java -jar ./zipkin-server/target/zipkin-server-*exec.jar
```

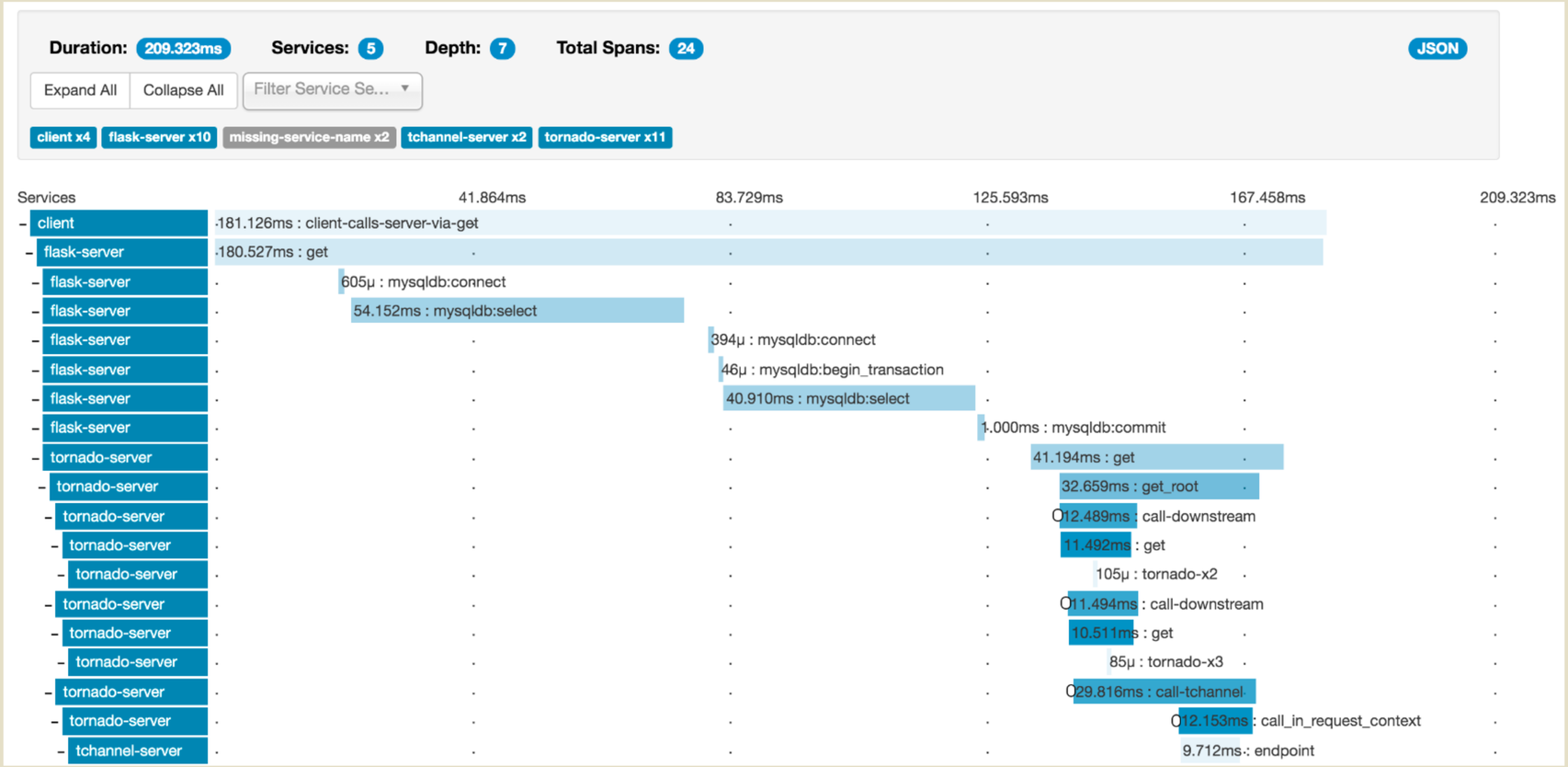
open http://localhost:9411/

(or `docker run --env STORAGE_TYPE=cassandra -d -p 9411:9411 openzipkin/zipkin`)

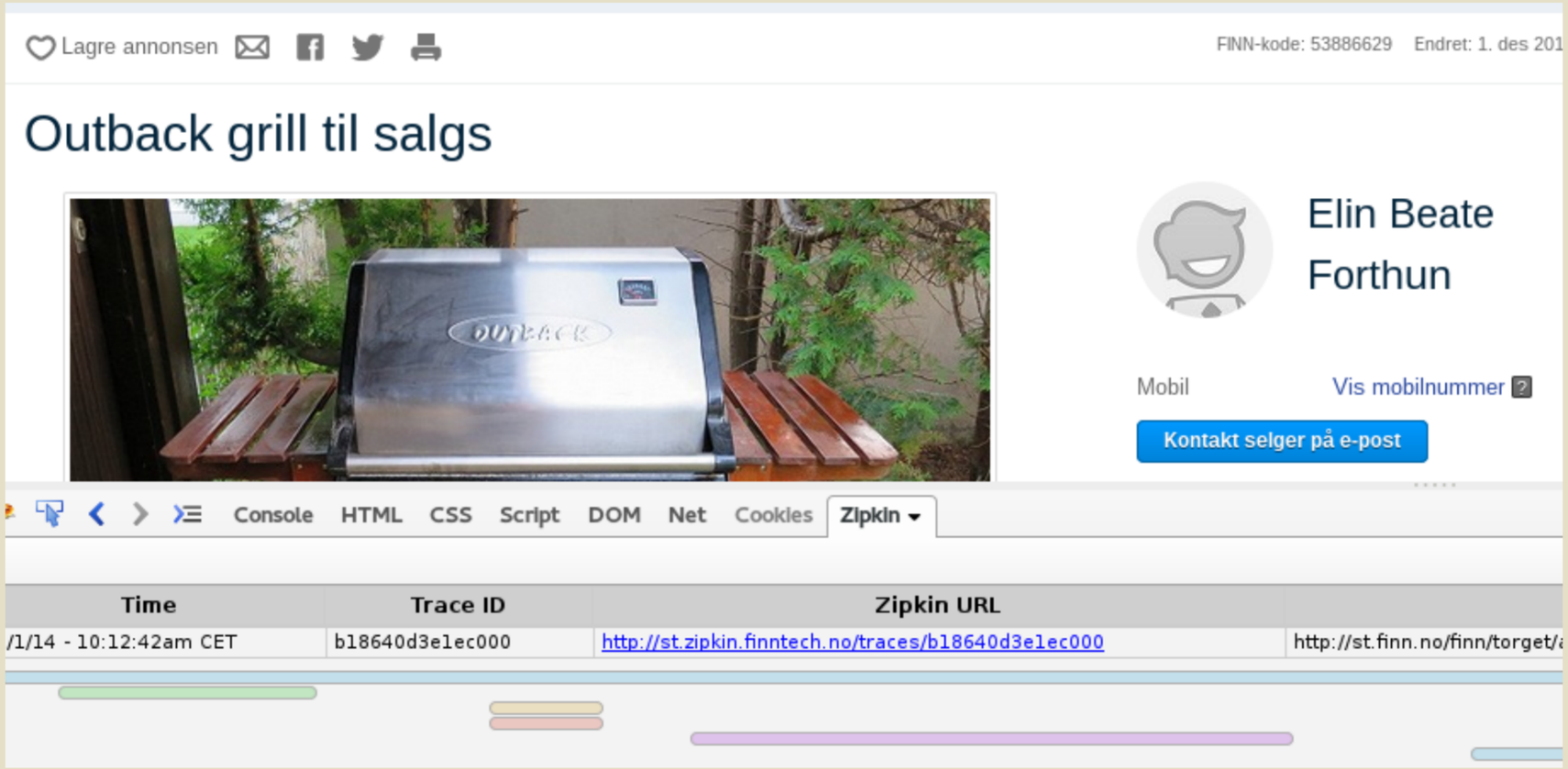
SEARCH TRACES



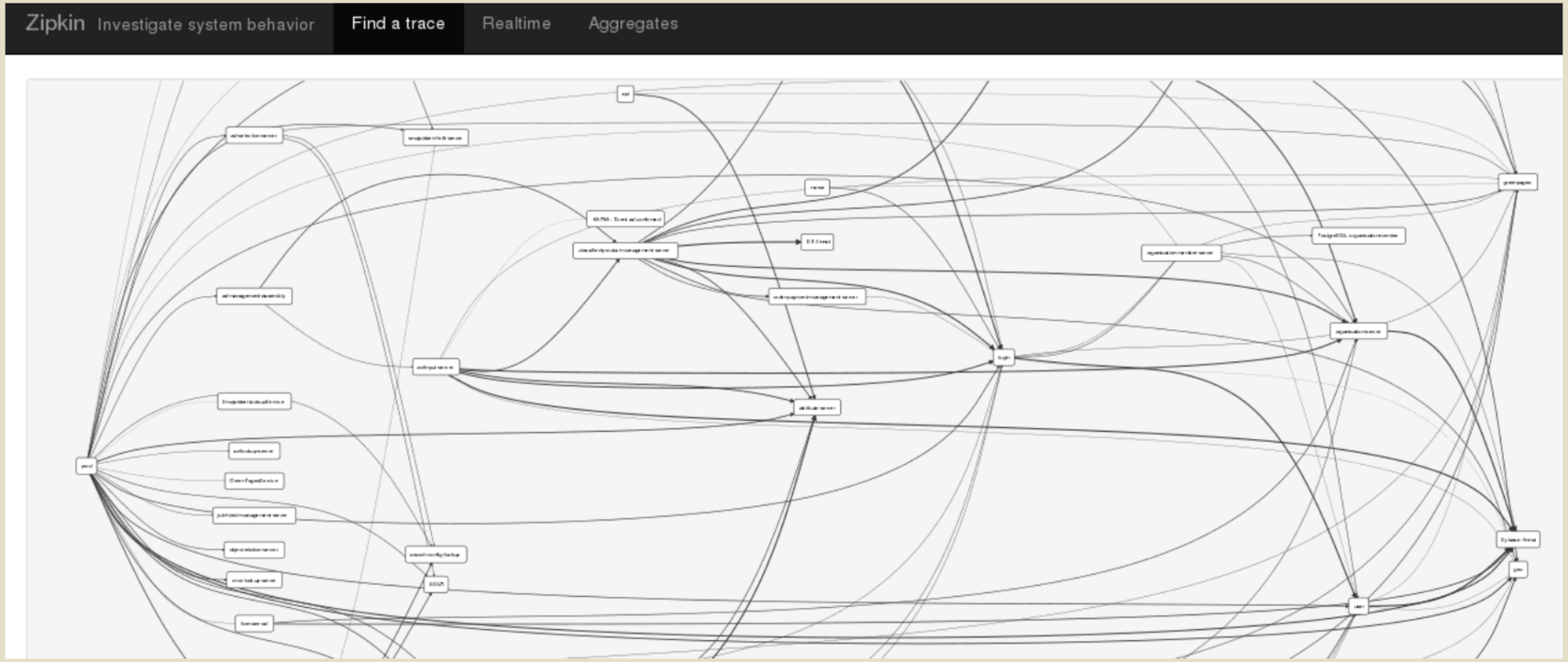
ANALYZE ONE TRACE



REALTIME IN BROWSER



PLATFORM CALL GRAPH



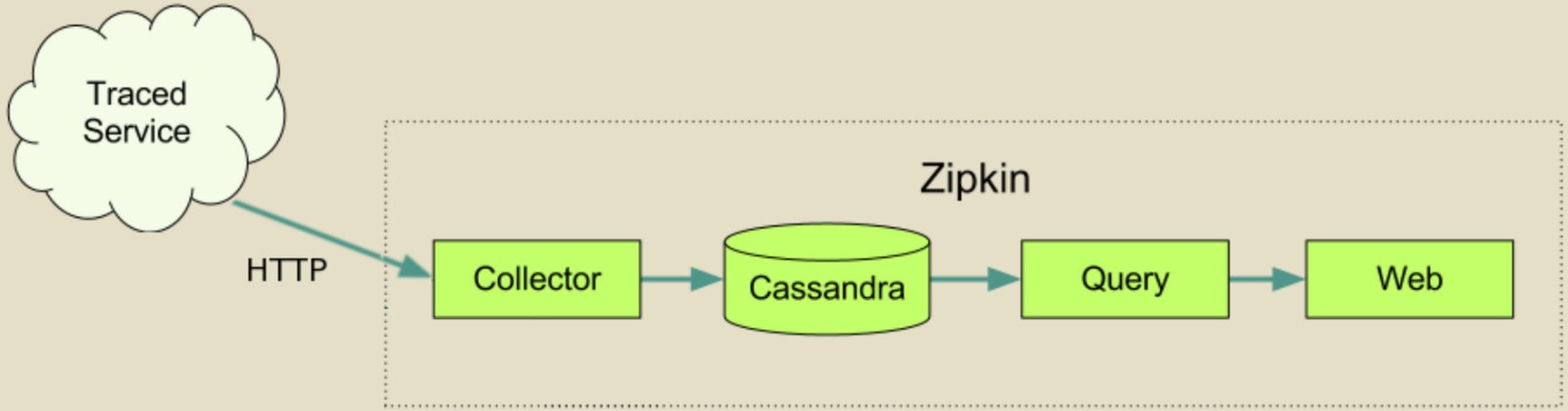
CLIENT | SERVER

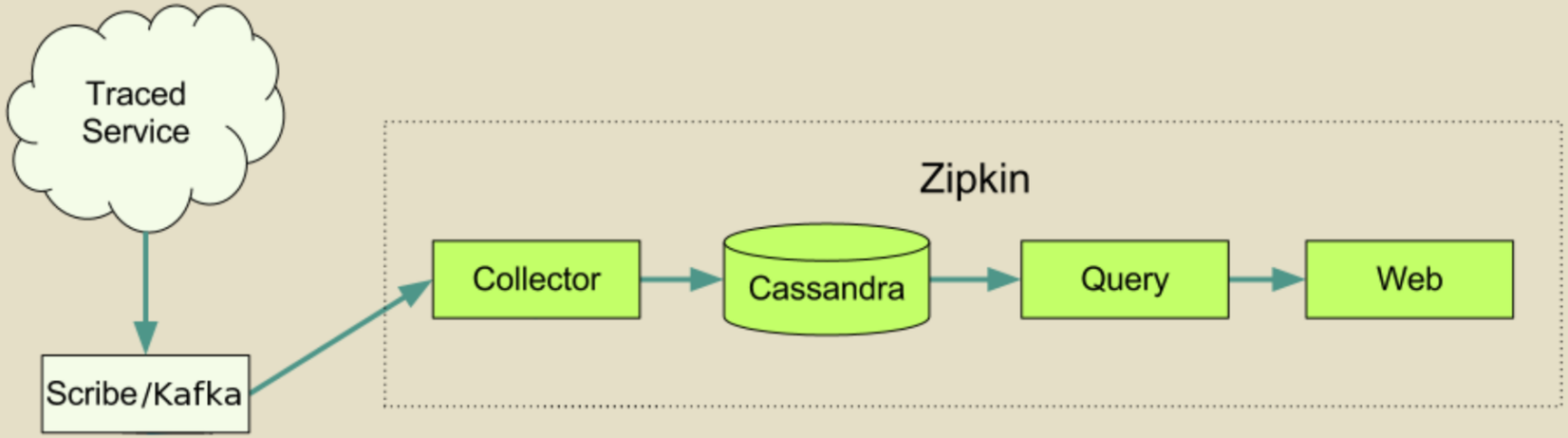
CS -->

--> SR

<-- SS

CR <--


Traced
Service
HTTP
Zipkin
Collector
Cassandra
Query
Web


Traced
Service
Scribe/Kafka
Zipkin
Collector
Cassandra
Query
Web

EXISTING INSTRUMENTATIONS

Language	Library	Framework
JavaScript	zipkin-js	cujoJS, express, restify
Python	pyramid_zipkin	Pyramid
Java	brave	Jersey, RestEASY, JAXRS2, Apache HttpClient, Mysql
Java	htrace	HDFS, HBase
Java	Spring Cloud Sleuth	Spring, Spring Cloud (e.g. Stream, Netflix)
Java	cassandra-zipkin-tracing	Apache Cassandra
Scala	finagle-zipkin	Finagle
Ruby	zipkin-tracer	Rack
C#	ZipkinTracerModule	OWIN, HttpHandler
Go	go-zipkin	x/net Context
Go	Go kit	Go kit
Scala	akka-tracing	Akka, Spray, Play
Java	Dropwizard Zipkin	Dropwizard
Java	Wingtips	Any Servlet API framework, roll-your-own, async framework support

SIMPLE HTTP CALL

```
Result result = doRequest(request);
```

SIMPLE HTTP CALL

```
SpanId spanId = clientTracer.startNewSpan(request.getRequestLine().getUri());

clientTracer.setClientSent();
Result result = doRequest(request);
clientTracer.setClientReceived();
```

(Brave-3.10 – https://github.com/openzipkin/brave)

SIMPLE C* CALL

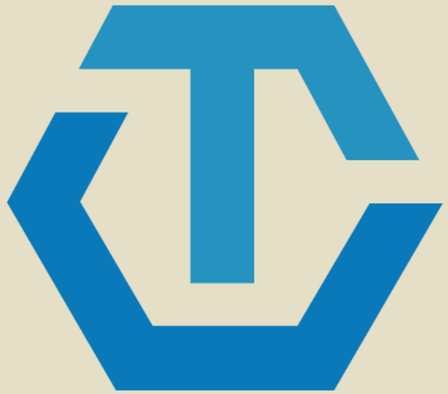
```
ResultSet result = session.execute(statement);
```

SIMPLE C* CALL

```
SpanId spanId = clientTracer.startNewSpan(statement.toString());
clientTracer.setClientSent();
ResultSet result = session.execute(statement);
clientTracer.setClientReceived();
```

HTTP CALL PASSING THROUGH HEADERS

```
SpanId spanId = clientTracer.startNewSpan(request.getRequestLine().getUri());

request.addHeader(Sampled.getName(), "1");
request.addHeader(TraceId.getName(), IdConversion.convertToString(spanId.traceId));
request.addHeader(SpanId.getName(), IdConversion.convertToString(spanId.spanId));

clientTracer.setClientSent();
Result result = doRequest(request);
clientTracer.setClientReceived();
```



OPENTRACING

SIMPLE HTTP CALL

```
Result result = doRequest(request);
```

SIMPLE HTTP CALL

```
try (Span span = tracer.buildSpan(request.getUri()).start()) {
    return doRequest(request);
}
```

(opentracing-java-0.14.0 – https://github.com/opentracing/opentracing-java)

SIMPLE C* CALL

```
ResultSet result = session.execute(statement);
```

SIMPLE C* CALL

```
try (Span span = tracer.buildSpan(statement.toString()).start()) {
    result = session.execute(statement);
}
```

HTTP CALL PASSING THROUGH HEADERS

```
try (Span span = tracer.buildSpan(request.getUri()).start()) {
    tracer.inject(span, request);
    return doRequest(request);
}
```

TRACING IN C*

Cassandra / CASSANDRA-1123

Allow tracing query details

Agile Board

Details

Type:	New Feature	Status:	RESOLVED
Priority:	Major	Resolution:	Fixed
Component/s:	Core	Fix Version/s:	1.2.0 beta 1
Labels:	None		

Description

In the spirit of ~~CASSANDRA-511~~, it would be useful to tracing on queries to see where latency is coming from: how long did row cache lookup take? key search in the index? merging the data from the sstables? etc.

We don't need to be as sophisticated as the techniques discussed in the following papers but they are interesting reading:

```
Tracing session: a55f4810-57aa-11e5-b0df-e700f669bcfc

 activity                                                                                                                                                  | timestamp                  | source    | source_elapsed
-----------------------------------------------------------------------------------------------------------------------------------------------------------+----------------------------+-----------+----------------
                                                                                                                                      Execute CQL3 query | 2015-09-10 12:56:53.521000 | 127.0.0.1 |              0
                                                  Parsing select * from vortex_powervomit_test.envelopes limit 10; [SharedPool-Worker-1] | 2015-09-10 12:56:53.523000 | 127.0.0.1 |           1635
                                                                                                                 Preparing statement [SharedPool-Worker-1] | 2015-09-10 12:56:53.526000 | 127.0.0.1 |           4318
                                                                                                                Computing ranges to query [SharedPool-Worker-1] | 2015-09-10 12:56:53.528000 | 127.0.0.1 |           7230
 Submitting range requests on 257 ranges with a concurrency of 1 (24892.65 rows per range expected) [SharedPool-Worker-1] | 2015-09-10 12:56:53.532000 | 127.0.0.1 |          10778
    Executing seq scan across 7 sstables for (min(-9223372036854775808), min(-9223372036854775808)] [SharedPool-Worker-3] | 2015-09-10 12:56:53.536000 | 127.0.0.1 |          14541
                                                 Submitted 1 concurrent range requests covering 257 ranges [SharedPool-Worker-1] | 2015-09-10 12:56:53.537000 | 127.0.0.1 |          16240
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.541000 | 127.0.0.1 |          19963
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.543000 | 127.0.0.1 |          22097
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.546000 | 127.0.0.1 |          24961
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.549000 | 127.0.0.1 |          27716
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.551000 | 127.0.0.1 |          29473
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.552000 | 127.0.0.1 |          31248
                                                        Read 11 live and 0 tombstone cells [SharedPool-Worker-3] | 2015-09-10 12:56:53.555000 | 127.0.0.1 |          33911
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.557000 | 127.0.0.1 |          36066
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.562000 | 127.0.0.1 |          40263
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.564000 | 127.0.0.1 |          42801
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.573000 | 127.0.0.1 |          51778
                                                       Seeking to partition beginning in data file [SharedPool-Worker-3] | 2015-09-10 12:56:53.579000 | 127.0.0.1 |          58210
                                                        Read 1 live and 0 tombstone cells [SharedPool-Worker-3] | 2015-09-10 12:56:53.582000 | 127.0.0.1 |          60590
                                                            Scanned 1 rows and matched 1 [SharedPool-Worker-3] | 2015-09-10 12:56:53.584000 | 127.0.0.1 |          62526
                                                                                                                      Request complete | 2015-09-10 12:56:53.588856 | 127.0.0.1 |          67856
```

```
   CO-ORDINATOR NODE                          REPLICA NODE


-->
   newSession(..)
       begin(..)
       trace(..)
       trace(..)
                                        --> initialiseMessage(..)
                                               trace(..)
                                               trace(..)
                                        <--
       trace(..)
   stopSession(..)
<--
```

ZIPKIN *IN* C*

- visualization
- detailed timings
- hierarchy and asynchronisity
- zero tracing overhead

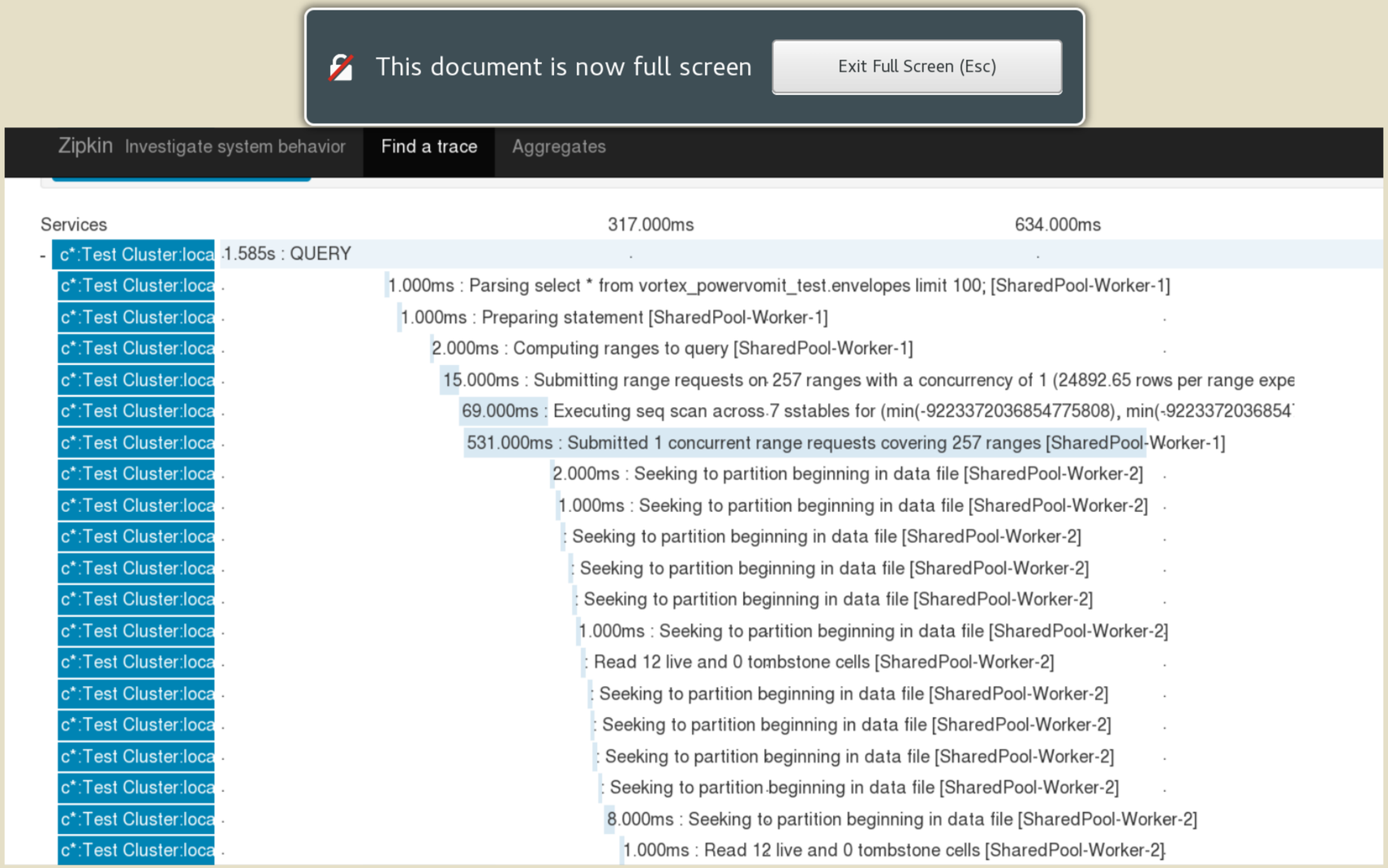
Cassandra-3.4

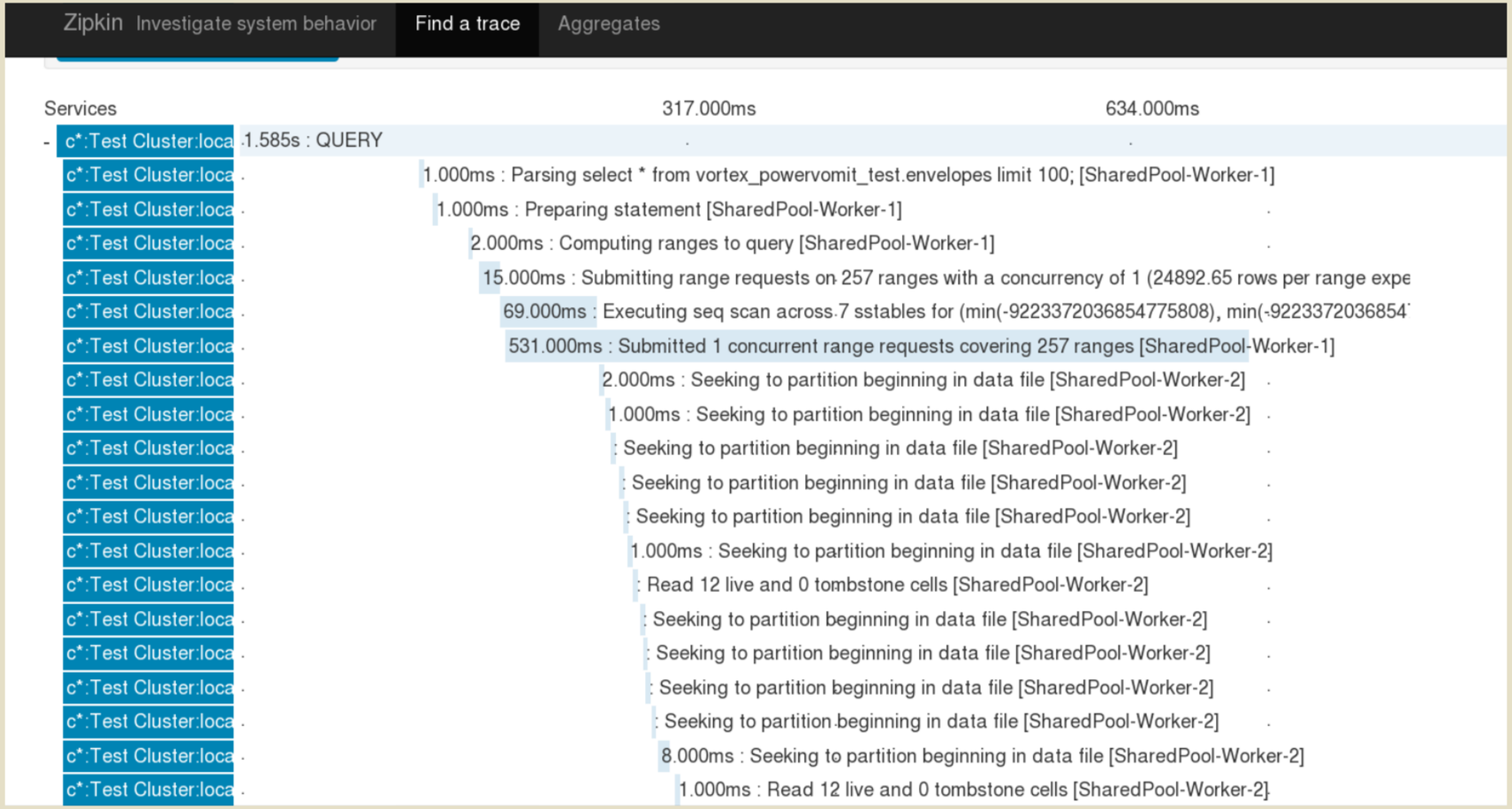
DOWNLOAD CASSANDRA-ZIPKIN-TRACING

```
git clone git@github.com:thelastpickle/cassandra-zipkin-tracing
cd cassandra-zipkin-tracing

# it's in the readme
mvn install
cp target/cassandra-zipkin-tracing-*.jar $CASSANDRA_HOME/lib/
cp lib/*  $CASSANDRA_HOME/lib/
```

then run, enabling zipkin tracing

```
JVM_OPTS= \
  "-Dcassandra.custom_tracing_class=com.thelastpickle.cassandra.tracing.ZipkinTracing" \
  bin/cassandra
```


This document is now full screen
Exit Full Screen (Esc)
Zipkin Investigate system behavior
Find a trace
Aggregates
Services
317.000ms
634.000ms
- c*:Test Cluster:loca 1.585s : QUERY
c*:Test Cluster:loca 1.000ms : Parsing select * from vortex_powervomit_test.envelopes limit 100; [SharedPool-Worker-1]
c*:Test Cluster:loca 1.000ms : Preparing statement [SharedPool-Worker-1]
c*:Test Cluster:loca 2.000ms : Computing ranges to query [SharedPool-Worker-1]
c*:Test Cluster:loca 15.000ms : Submitting range requests on 257 ranges with a concurrency of 1 (24892.65 rows per range expe
c*:Test Cluster:loca 69.000ms : Executing seq scan across 7 sstables for (min(-9223372036854775808), min(-9223372036854
c*:Test Cluster:loca 531.000ms : Submitted 1 concurrent range requests covering 257 ranges [SharedPool-Worker-1]
c*:Test Cluster:loca 2.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 1.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 1.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Read 12 live and 0 tombstone cells [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 8.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 1.000ms : Read 12 live and 0 tombstone cells [SharedPool-Worker-2]

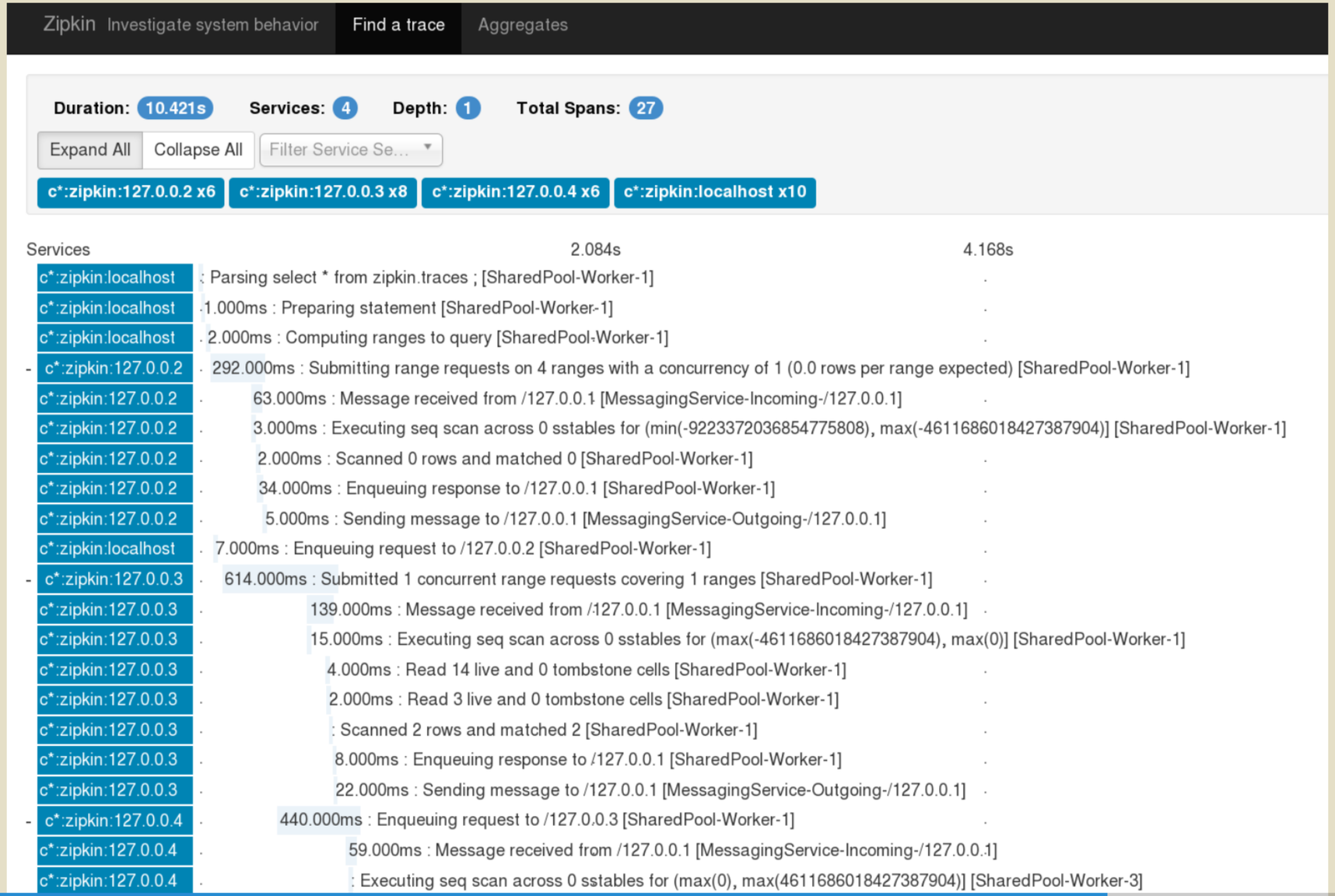

Zipkin Investigate system behavior
Find a trace
Aggregates
Services
317.000ms
634.000ms
- c*:Test Cluster:loca 1.585s : QUERY
c*:Test Cluster:loca 1.000ms : Parsing select * from vortex_powervomit_test.envelopes limit 100; [SharedPool-Worker-1]
c*:Test Cluster:loca 1.000ms : Preparing statement [SharedPool-Worker-1]
c*:Test Cluster:loca 2.000ms : Computing ranges to query [SharedPool-Worker-1]
c*:Test Cluster:loca 15.000ms : Submitting range requests on 257 ranges with a concurrency of 1 (24892.65 rows per range expe
c*:Test Cluster:loca 69.000ms : Executing seq scan across 7 sstables for (min(-9223372036854775808), min(-9223372036854
c*:Test Cluster:loca 531.000ms : Submitted 1 concurrent range requests covering 257 ranges [SharedPool-Worker-1]
c*:Test Cluster:loca 2.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 1.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 1.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Read 12 live and 0 tombstone cells [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 8.000ms : Seeking to partition beginning in data file [SharedPool-Worker-2]
c*:Test Cluster:loca 1.000ms : Read 12 live and 0 tombstone cells [SharedPool-Worker-2]

ZIPKIN *ACROSS* C*

ZIPKIN *ACROSS* C*

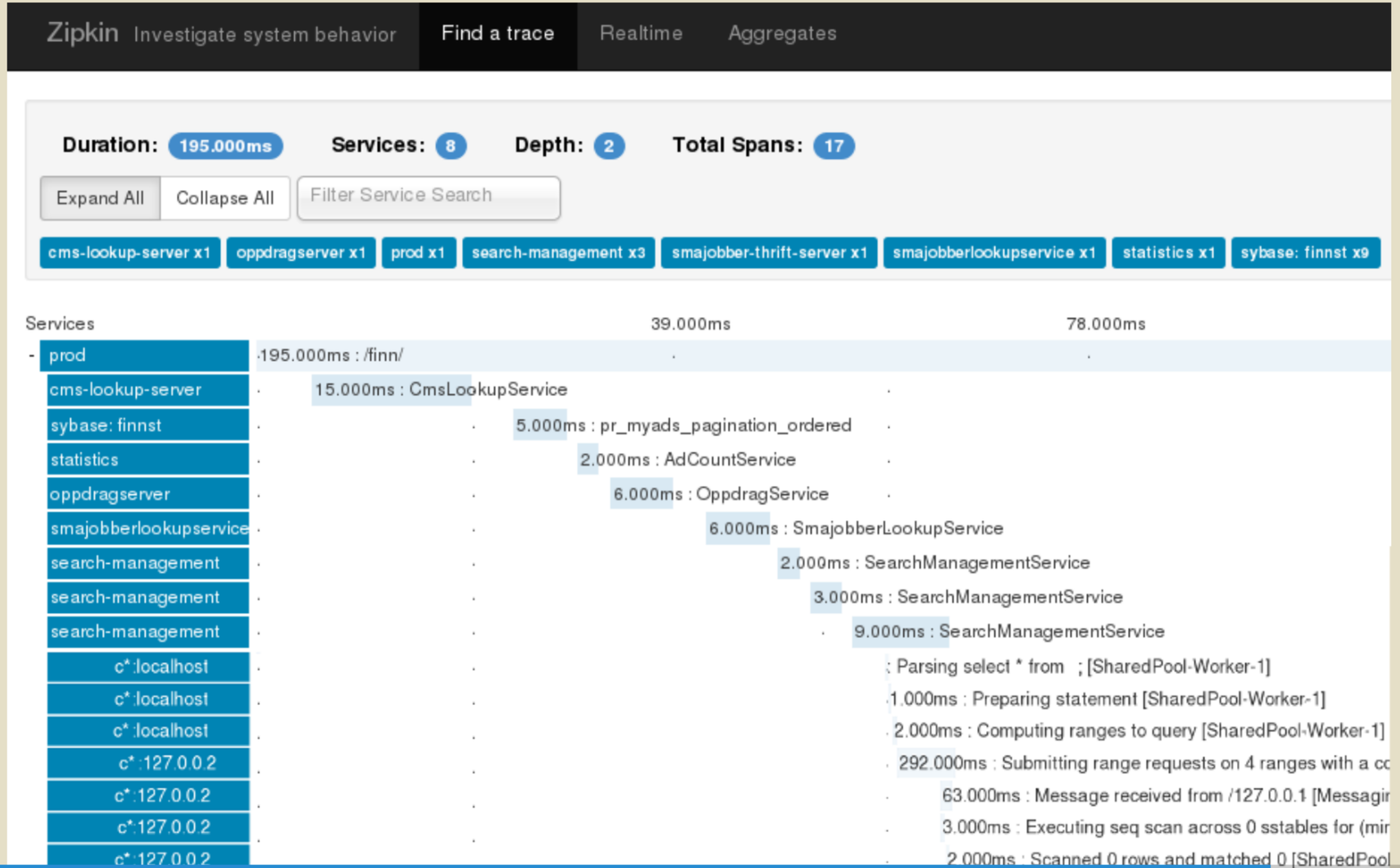
```
   CO-ORDINATOR NODE                          REPLICA NODE


-->
   newSession(..)
       begin(..)
       trace(..)
       trace(..)
                   (zipkin headers) --> initialiseMessage(..)
                                            trace(..)
                                            trace(..)
                                    <--
       trace(..)
   stopSession(..)
<--
```

ZIPKIN *ACROSS* C*



ZIPKIN *INTO* C*

ZIPKIN *INTO* C*



HTTP CALL PASSING THROUGH HEADERS

```
SpanId spanId = clientTracer.startNewSpan(request.getRequestLine().getUri());

request.addHeader(Sampled.getName(), "1");
request.addHeader(TraceId.getName(), IdConversion.convertToString(spanId.traceId));
request.addHeader(SpanId.getName(), IdConversion.convertToString(spanId.spanId));

clientTracer.setClientSent();
Result result = doRequest(request);
clientTracer.setClientReceived();
```

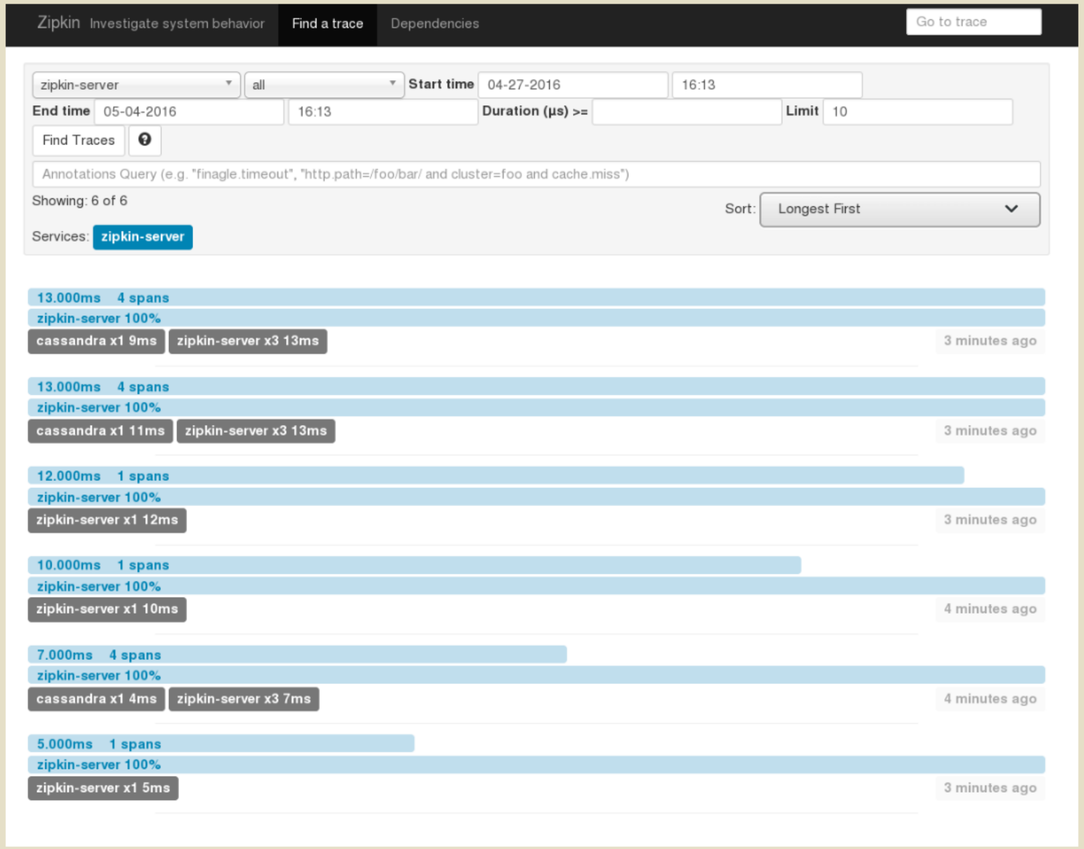
C* CALL USING CUSTOM PAYLOAD

```
SpanId spanId = clientTracer.startNewSpan(statement.toString());

ByteBuffer traceHeaders = ByteBuffer.wrap(spanId.bytes());
statement.setOutgoingPayload(singletonMap("zipkin", traceHeaders));

clientTracer.setClientSent();
ResultSet result = session.execute(statement);
clientTracer.setClientReceived();
```

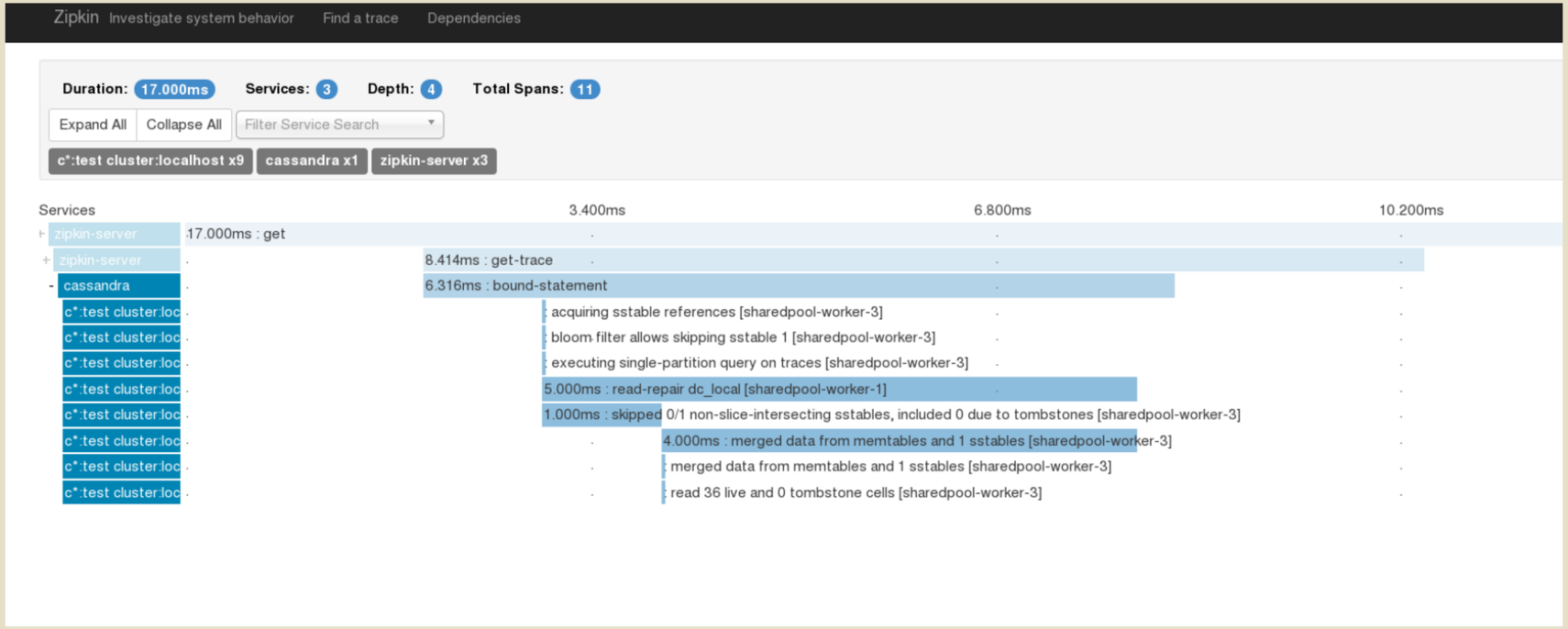
enable zipkin tracing *and* the custom payload handler

```
bin/cassandra
    -Dcassandra.custom_tracing_class=..ZipkinTracing
    -Dcassandra.custom_query_handler_class=..CustomPayloadMirroringQueryHandler
```

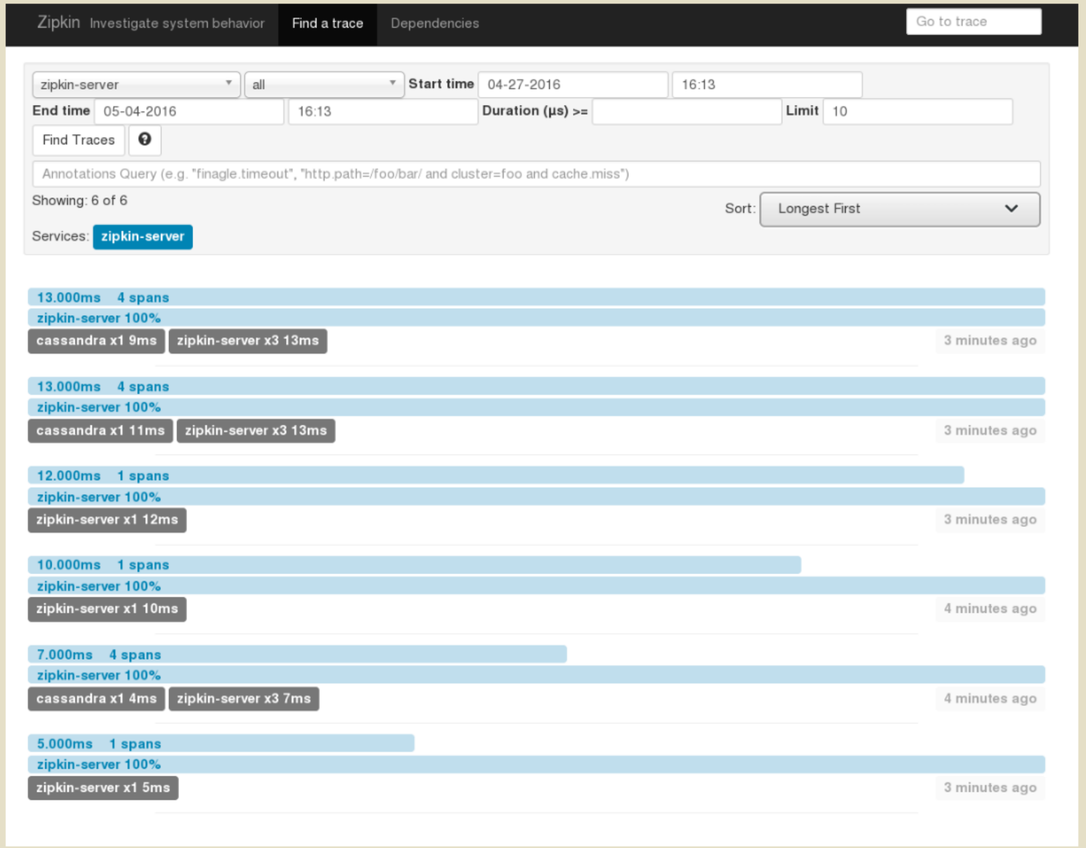
ZIPKIN SELF TRACING



ANALYZE ONE TRACE



ZIPKIN ON CASSANDRA-3.9



SO MUCH AWESOMENESS

- **UDT**
- **MATERIALIZED VIEW**
- **SASI**
- **TimeWindowCompactionStrategy**

SO MUCH AWESOMENESS

7 Tables --> 2 Tables
1.5k LOC removed
Performance!

THANKS

- Zipkin – github.com/openzipkin/zipkin
- Brave (zipkin java instrumentation) – openzipkin-brave
- OpenTracing (agnostic instrumentation) – opentracing-zipkin

- Zipkin Cassandra implementation (& this presentation)
  – github.com/thelastpickle/cassandra-zipkin-tracing

Mick Semb Wever

TWITTER @mck_sw
mick@thelastpickle.com

THE LAST PICKLE

APACHE CASSANDRA CONSULTING & SERVICES