ARE WE ALL ON THE SAME PAGE?

LET'S FIX THAT

Luis Mineiro @voidmaze
SRE @ Zalando
SREcon EMEA 2019
ZALANDO AT A GLANCE

~ 5.4 billion EUR revenue 2018

> 15,500 employees in Europe

> 80% of visits via mobile devices

> 300 million visits per month

> 400,000 product choices

> 27 million active customers

~ 2,000 brands

17 countries

as of October 2019
as of October 2019

ZALANDO OFFICES

1. BERLIN HEADQUARTERS
2. ERFURT TECH OFFICE
3. MÖNCHENGLADBACH TECH OFFICE
4. DORTMUND TECH HUB
5. DUBLIN TECH HUB
6. HELSINKI TECH HUB
7. HAMBURG ADTECH LAB
THE AGE OF THE MONOLITH

Single, large boxes that did everything

Jimmy
The Monolith

Request
Response
MONITORING THE MONOLITH

Ops Monitoring

- Is the box alive?
- Is the monolith process up?

Devs Monitoring

- Are requests returning errors?
- Are requests reasonably fast?

Photo by Deneen LT on Pexels
MODERN MICROSERVICES ARCHITECTURES

Amazon internal service dependency visualization
EXAMPLE - PLACING AN ORDER

Customer → Web Frontend → Checkout Service → Payment Gateway → Payment Service

- Typical Payment Blackbox
- Another Shady Service

- Risk Service
- A Queue of Sorts
- Order Service

- Logistics Service → Stock Reservation Service
- Coupon Service
- Machine Learning Shenanigans
- Random BI Service → Accounting Service
"DevOps" Monitoring

- Is the box alive?
- Is the micro-service process up?
- Are requests returning errors?
- Are requests reasonably fast?
FAILURE PLACING AN ORDER

Customer → Web Frontend → Checkout Service → Payment Gateway → Payment Service

- Typical Payment Blackbox
- Another Shady Service

- Risk Service
- A Queue of Sorts
- Order Service

- Logistics Service
- Stock Reservation Service
- Coupon Service
- Machine Learning Shenanigans
- Random BI Service

Accounting Service
ALERTS ON FAILURE PLACING AN ORDER

Customer ➔ Web Frontend ➔ Checkout Service ➔ Payment Gateway ➔ Payment Service ➔ Typical Payment Blackbox ➔ Another Shady Service

- Risk Service
- A Queue of Sorts ➔ Order Service
- Logistics Service ➔ Stock Reservation Service ➔ Coupon Service ➔ Machine Learning Shenanigans ➔ Random BI Service ➔ Accounting Service
ALERTS ON FAILURE PLACING AN ORDER

Customer → Web Frontend → Checkout Service → Payment Gateway → Payment Service → Typical Payment Blackbox → Another Shady Service

Risk Service → A Queue of Sorts → Order Service

Logistics Service → Stock Reservation Service

Coupon Service

Machine Learning Shenanigans

Random BI Service → Accounting Service

Photo by Antoine Plüss on Unsplash
Good signal to noise ratio.
Create an alert rule "here"
ALERT ON THE SYMPTOM

Customer

- Web Frontend
  - Checkout Service
    - Payment Gateway
      - Payment Service
        - Typical Payment Blackbox
          - Another Shady Service
            - Logistics Service
              - Stock Reservation Service
                - Machine Learning Shenanigans
              - Coupon Service
                - Random BI Service
            - Accounting Service
          - A Queue of Sorts
            - Order Service
              - Risk Service
Single alert triggered

Customer → Web Frontend → Checkout Service → Payment Gateway → Payment Service

- Risk Service
- A Queue of Sorts
- Order Service

- Typical Payment Blackbox
- Another Shady Service

- Logistics Service
- Stock Reservation Service
- Coupon Service
- Machine Learning Shenanigans
- Random BI Service

Alert on the Symptom

[Diagram showing flow of services and alert triggers]
ALERT ON THE SYMPTOM - DIFFERENT ISSUE

Customer → Web Frontend → Checkout Service → Payment Gateway → Payment Service

- Typical Payment Blackbox
- Another Shady Service

- Risk Service
- A Queue of Sorts
- Order Service

- Logistics Service
- Stock Reservation Service
- Coupon Service
- Machine Learning Shenanigans
- Random BI Service
- Accounting Service
ALERT ON THE SYMPTOM - DIFFERENT ISSUE

Single alert triggered
PLACING AN ORDER - ALERT BOMBING

Customer

Web Frontend → Checkout Service → Payment Gateway → Payment Service

Typical Payment Blackbox → Another Shady Service

Risk Service

A Queue of Sorts

Order Service

Logistics Service

Stock Reservation Service

Coupon Service

Machine Learning Shenanigans

Random BI Service

Accounting Service

Single alert triggered

👤 Customer

⚠ Machine Learning Shenanigans
ALERTING FOR MICROSERVICES

Charity Majors
@mipsytipsy

alright, this is a damn good question. and tbh i am surprised it doesn't come up more often, because it gets right to the beating heart of what makes any microservices architecture good or bad.

Jacob @jhscoot
In a 'microservices organization' where teams own specific components/services of a distributed production system, who is responsible for triage/debugging/routing of issues that don't present with a clear owner? And how do they not hate their lives?
@mipsytipsy any thoughts?

293  6:43 AM - Apr 24, 2019

106 people are talking about this
Adaptive Paging is an alert handler that leverages the causality from tracing and OpenTracing's semantic conventions to page the team closest to the problem.
DISTRIBUTED TRACING AND OPENTRACING

- A trace tells the **story of a transaction or workflow as it propagates** through a distributed system.

- It's basically a directed acyclic graph (DAG), with a **clear start** and a **clear end** - no loops.

- A trace is made up of **spans** representing contiguous segments of work in that trace.

- Opentracing is a set of **vendor-neutral APIs** and code instrumentation **standard for distributed tracing**
DISTRIBUTED TRACING AND OPENTRACING OPENTELEMETRY

- A trace tells the **story of a transaction or workflow as it propagates** through a distributed system.
- It's basically a directed acyclic graph (DAG), with a **clear start** and a **clear end** - no loops.
- A trace is made up of **spans** representing contiguous segments of work in that trace.
- OpenTelemetry is made up of an integrated set of APIs and libraries as well as a collection mechanism via an agent and collector. It also does **distributed tracing**

[Diagram: OT + Tracing = OpenTelemetry]
**Span**: a named operation which records the **duration**, usually a remote procedure call, with optional **Tags** and Logs.
**Tag**: A "mostly" arbitrary **Key:Value pair** (value can be a string, number or bool)

### OPENTRACING CONCEPTS

<table>
<thead>
<tr>
<th>Service &amp; Operation</th>
<th>0ms</th>
<th>53.6ms</th>
<th>107.19ms</th>
<th>160.79ms</th>
<th>214.38ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>web-frontendl render_checkout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>checkout-service place_order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>payment-gw take_payment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>119.89ms</td>
</tr>
<tr>
<td>risk-service evaluate_risk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4.08ms</td>
</tr>
<tr>
<td>queue-of-sorts push_order_event</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.53ms</td>
</tr>
<tr>
<td>order-service accept_order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>82.56ms</td>
</tr>
</tbody>
</table>

![Diagram showing tags and values](image)
**OPENTRACING SEMANTIC CONVENTIONS**

<table>
<thead>
<tr>
<th>Span tag name</th>
<th>Type</th>
<th>Notes and examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>component</td>
<td>string</td>
<td>The <strong>software package</strong>, framework, library, or module that generated the associated Span. E.g., &quot;checkout-service&quot;.</td>
</tr>
<tr>
<td>error</td>
<td>bool</td>
<td><strong>true</strong> if and only if the application considers the operation represented by the Span to have failed</td>
</tr>
<tr>
<td>peer.service</td>
<td>string</td>
<td><strong>Remote service name</strong> (for some unspecified definition of &quot;service&quot;). E.g., &quot;accounting-service&quot;</td>
</tr>
<tr>
<td>span.kind</td>
<td>string</td>
<td>Either &quot;client&quot; or &quot;server&quot; for the appropriate roles in an RPC.</td>
</tr>
<tr>
<td>... and more</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Openetracing semantic conventions*
OPENTRACING MONITORING SIGNALS

The Four Golden Signals

Failed operation (error=true)

Latency
ERROR RATE ALERTING RULE

component: checkout_service && operation: place_order

Alert triggered.
## ALERT PAYLOAD

### 1. web-fronted: render_checkout
- Transaction ID: 7c22faa
- Duration: 214.38ms
- Spans: 17
- Errors: 3
- Services:
  - accounting-service: 1
  - checkout-service: 1
  - coupon-service: 1
  - logistics-service: 1
  - machine-learning-crap: 1
  - order-service: 2
  - payment-blackbox: 1
  - payment-gw: 1
  - payment-service: 1
  - queue-of-sorts: 1
  - random-bi-svc: 1
  - risk-service: 1
  - shady-service: 1
  - stock-reservation-svc: 2
- Timestamp: May 10, 2:40:50 pm
- Days ago: 2

### 2. web-fronted: render_checkout
- Transaction ID: 7f72506
- Duration: 265.97ms
- Spans: 17
- Errors: 3
- Services:
  - accounting-service: 1
  - checkout-service: 1
  - coupon-service: 1
  - logistics-service: 1
  - machine-learning-crap: 1
  - order-service: 2
  - payment-blackbox: 1
  - payment-gw: 1
  - payment-service: 1
  - queue-of-sorts: 1
  - random-bi-svc: 1
  - risk-service: 1
  - shady-service: 1
  - stock-reservation-svc: 2
- Timestamp: May 10, 2:40:49 pm
- Days ago: 2

### 3. web-fronted: render_checkout
- Transaction ID: 50b1e32
- Duration: 288.33ms
- Spans: 17
- Errors: 3
- Services:
  - accounting-service: 1
  - checkout-service: 1
  - coupon-service: 1
  - logistics-service: 1
  - machine-learning-crap: 1
  - order-service: 2
  - payment-blackbox: 1
  - payment-gw: 1
  - payment-service: 1
  - queue-of-sorts: 1
  - random-bi-svc: 1
  - risk-service: 1
  - shady-service: 1
  - stock-reservation-svc: 2
- Timestamp: May 10, 2:40:40 pm
- Days ago: 2
1. Starting at the span which was defined as the signal - `place_order`
WALKING THROUGH A TRACE

1. Starting at the span which was defined as the signal - `place_order`
2. Inspect every child span's tags
3. Follow path with `error=true`
WALKING THROUGH A TRACE

1. Starting at the span which was defined as the signal - `place_order`
2. Inspect every child span's tags
3. Follow path with `error=true`
4. Rinse and repeat until no more children
ALERT ON THE SYMPTOM

Customer

Web Frontend

Checkout Service

Payment Gateway

Payment Service

Typical Payment Blackbox

Another Shady Service

Risk Service

A Queue of Sorts

Order Service

Logistics Service

Stock Reservation Service

Coupon Service

Machine Learning Shenanigans

Random BI Service

Accounting Service

🔍 Signal

👤
ALERT ON THE SYMPTOM

Customer ➔ Web Frontend ➔ Checkout Service ➔ Payment Gateway ➔ Payment Service ➔ Typical Payment Blackbox ➔ Another Shady Service ➔ Logistics Service ➔ Stock Reservation Service ➔ Coupon Service ➔ Machine Learning Shenanigans ➔ Random BI Service ➔ Accounting Service

Single page dispatched to the team operating the Accounting Service
ALERT ON THE SYMPTOM - DIFFERENT ISSUE

Customer → Web Frontend → Checkout Service → Payment Gateway → Payment Service

Signal:
- Risk Service
- A Queue of Sorts
- Order Service

Other Services:
- Another Shady Service
- Typical Payment Blackbox
- Logistics Service
- Stock Reservation Service
- Coupon Service
- Machine Learning Shenanigans
- Random BI Service
- Accounting Service
ALERT ON THE SYMPTOM - DIFFERENT ISSUE

Single page dispatched to the team operating the Payment Service

Customer ➔ Web Frontend ➔ Checkout Service ➔ Payment Gateway ➔ Payment Service

- Typical Payment Blackbox
- Another Shady Service

- Risk Service
- A Queue of Sorts
- Order Service

- Logistics Service
- Stock Reservation Service
- Coupon Service
- Machine Learning Shenanigans
- Random BI Service
- Accounting Service
Charity Majors
@mipsytipsy

ahhhh that's fucking smart. first i've heard of this. 😍

Luis Mineiro @voidmaze
Replying to @mipsytipsy

We're now addressing this with a custom alert handler that leverages the causality from tracing and Opentracing's semantic conventions to page the team closest to the problem
CHALLENGES

- Multiple child spans with error=true:
  - Follow each path, attribute the probable cause a score
  - Analyze more exemplars and adjust the scores
  - Worse case scenario, page both probable causes

- Missing instrumentation or circuit breaker open
  - Use the peer.service and span.kind=client tag to suggest which dependency would be the target

- Mapping services to escalation
  - Owning team may not have their own on-call escalation. Fallback to closest
THANK YOU

QUESTIONS?

Luis Mineiro @voidmaze

We're Hiring!

https://jobs.zalando.com