Hunting for Risky Dependencies
Geo Data SRE & Zero Outages

Geo SRE
Google Pay

Recent activity

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec 26, 2021</td>
<td>Starbucks</td>
<td>$2.86</td>
</tr>
<tr>
<td>Dec 18, 2021</td>
<td>Super Duper Burgers</td>
<td>$15.80</td>
</tr>
<tr>
<td>Nov 30, 2021</td>
<td>Starbucks</td>
<td>+ $2.00</td>
</tr>
</tbody>
</table>

Desired Reliability Attributes:
- Many 9s of Availability
- Low Latency
- Accessible globally
- Suitable data

GFE: Google Front End (load balancer)

Google Maps (Geo) Backends

GPay user: View Purchase
Google Pay

Correct dependency (not used):

Risky Dependency (used):

Geo Data

GFE: Google Front End (load balancer)

Actual Reliability Attributes:
- Few 9s of Availability
- Replicated in USA only
- Unaware of dependency
Google Business Profile (GBP) → Google Pay

- **GFE**: Google Front End (load balancer)
- **GBP**: Google Business Profile
- **GPay User**: Transaction History
- **Merchant User**: Update a business listing

**Geo Data**:
- Moderation System
- Canonical Store
- Serving Layer

A → B → C
What Should Be on the Critical Path?

**Geo Data**

- AFE 1
- Moderation System
- Canonical Store
- Serving Layer

---

**AFE 2**

- **Bolded / Red:** Not on the end-user path
- **Dashed Arrow:** Asynchronous flow

---

**GFE:** Google Front End (load balancer)

**AFE:** Application Front End

---

**Geo SRE**
How Isolated Are We Really?
Moderation Storage Errors

Geo Data

Merchant

GFE → GBP → Moderation System → Canonical Store → Serving Layer

Spanner RPC Errors per Status

Err/s

13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00 22:00 23:00 28 Feb
Major Outage in GBP!

Geo Data

Merchant

A

B

C

Moderation System

Canonical Store

Serving Layer

[Graph showing recent failure ratio for GBP]
Why does this happen?

RPC Tree

Dev Visibility
Finding these Risks
In Theory:
In Practice: Horizontal Monitoring

Externally-visible Edges:
GFE → A, A → C, C → D, D → E

Integrate OpenTelemetry

store metadata

SQL: filter by originator = "GFE"

Grey: Originator inside Google
Green: Originator outside Google
In Practice: Horizontal Monitoring

Externally-visible Edges:
GFE → A, A → C, C → D, D → E

Hand-curated Intents
External: A, B, C, E, D
Internal: A, B

Risky Dependencies:
C → D, D → E

Service owners state whether their service intends to be internal or external.

Filter by backends that intend to be internal-only
Audit results
Fixing these risks
Migrate

Make Optional

Deprecate
Results
33 Risky Dependencies

23 External Frontends

10 Internal Backends
To Recap

1. Invariant

Internal backends should not be on the critical path for end users.

2. Problem

These backends often become risky dependencies when service complexity increases.

3. Solution

We can find and fix these risky dependencies with OpenTelemetry and engineering work.

Questions? Comments?
pikle@google.com
Appendix
Additional Resources

- OpenTelemetry: [https://opentelemetry.io/](https://opentelemetry.io/)
- OpenTelemetry + Google Cloud: [https://cloud.google.com/learn/what-is-opentelemetry](https://cloud.google.com/learn/what-is-opentelemetry)
- Baggage: [https://opentelemetry.io/docs/concepts/signals/baggage/](https://opentelemetry.io/docs/concepts/signals/baggage/)