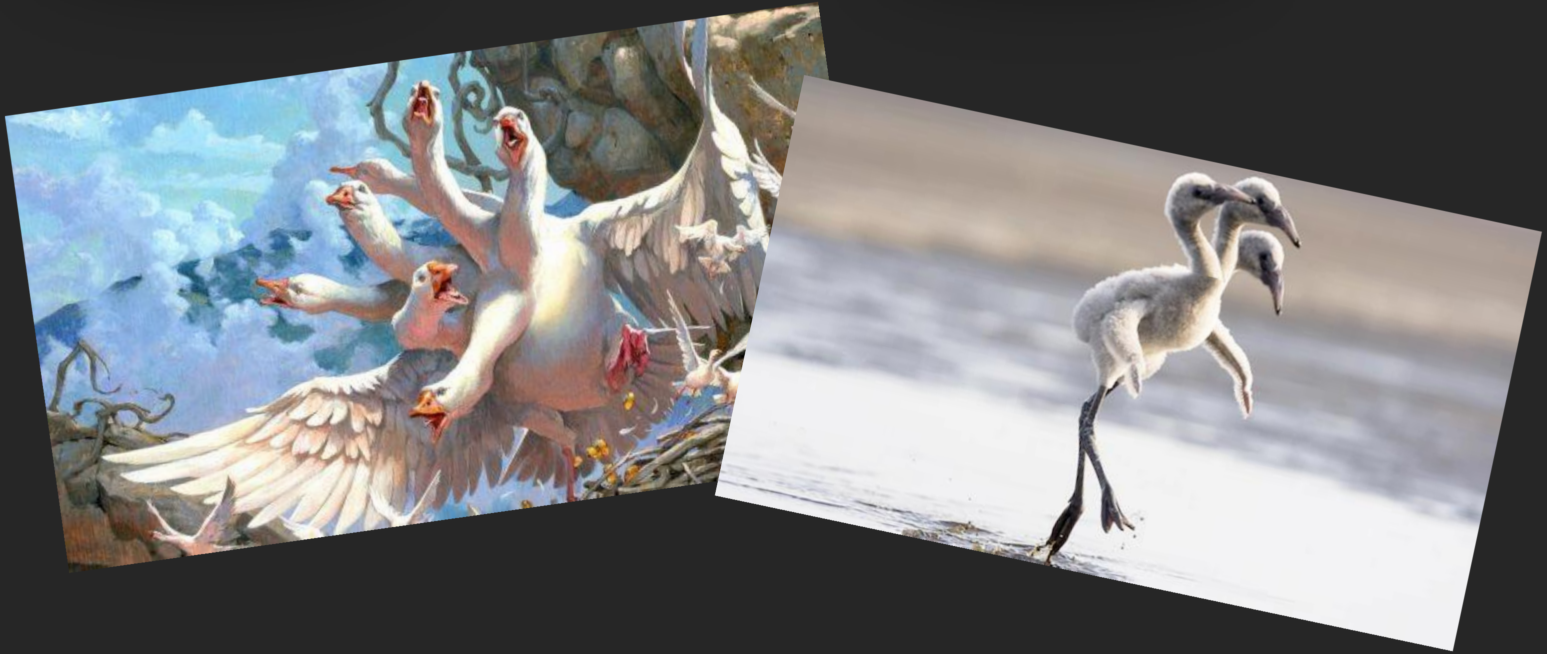
Fusion Fever

Building a Next-Level GraphQL Platform

Who

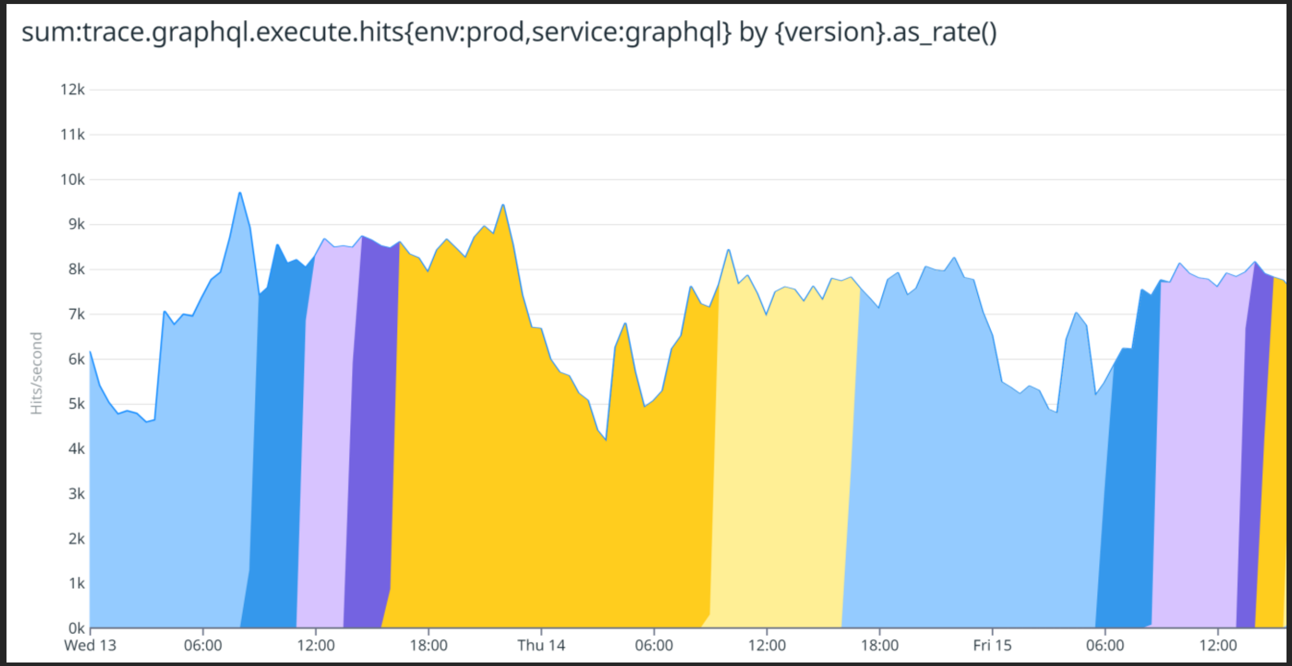
Tobias Tengler

Martin Disch



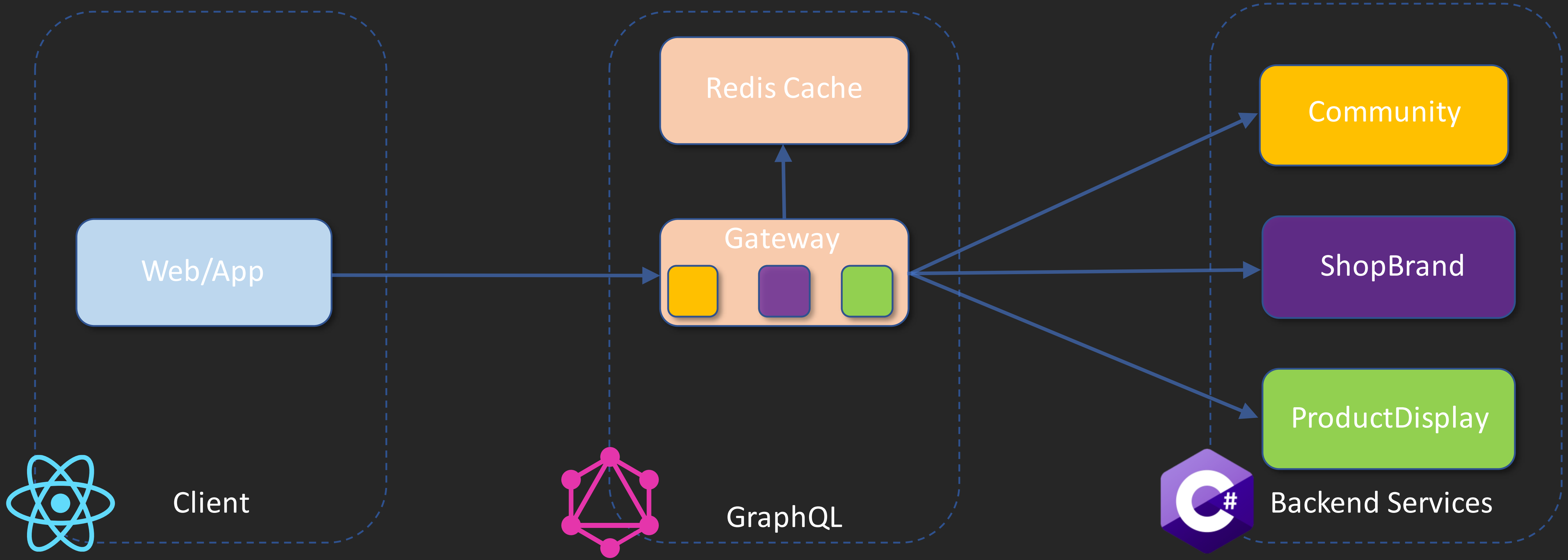
One year ago...

- Two of our colleagues told you about our GraphQL setup
- Currently does about 7k operations/s
- But is it good?

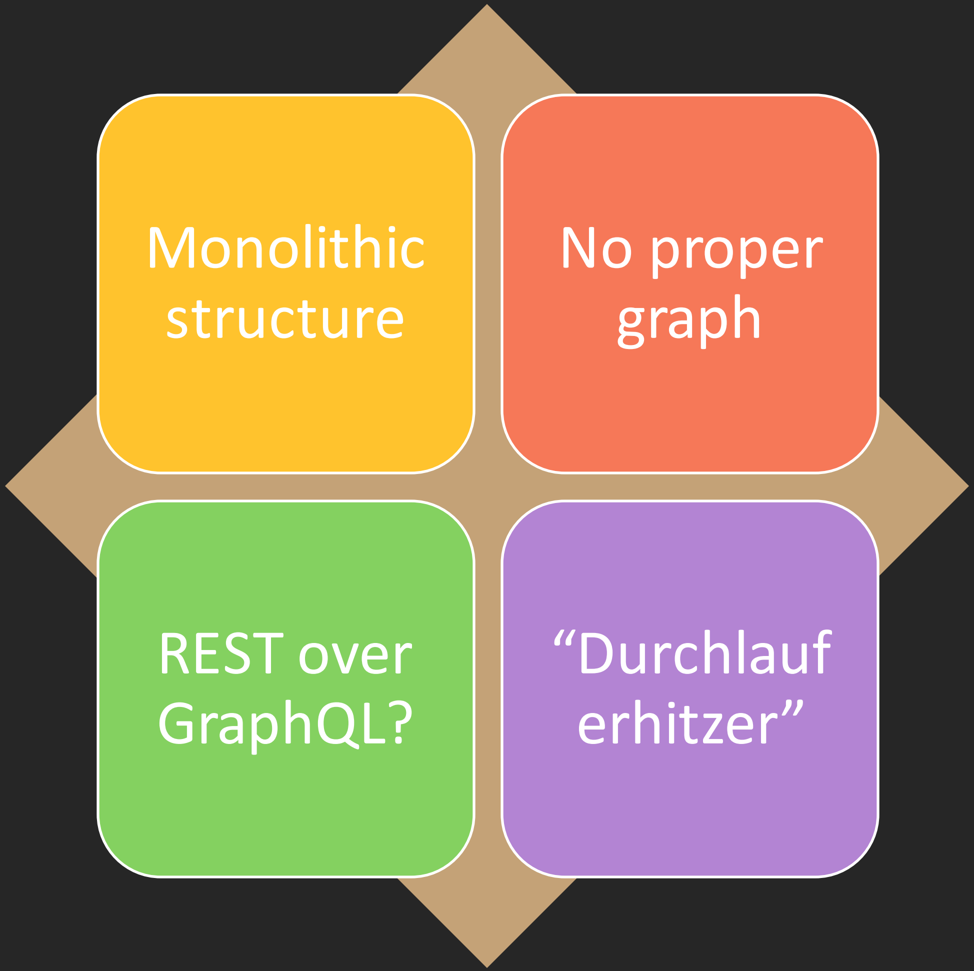


sum:trace.graphql.execute.hits{env:prod,service:graphql} by {version}.as_rate()

Situation


Redis Cache
Web/App
Gateway
Community
ShopBrand
ProductDisplay
Client
GraphQL
Backend Services

Issues



Monolithic structure

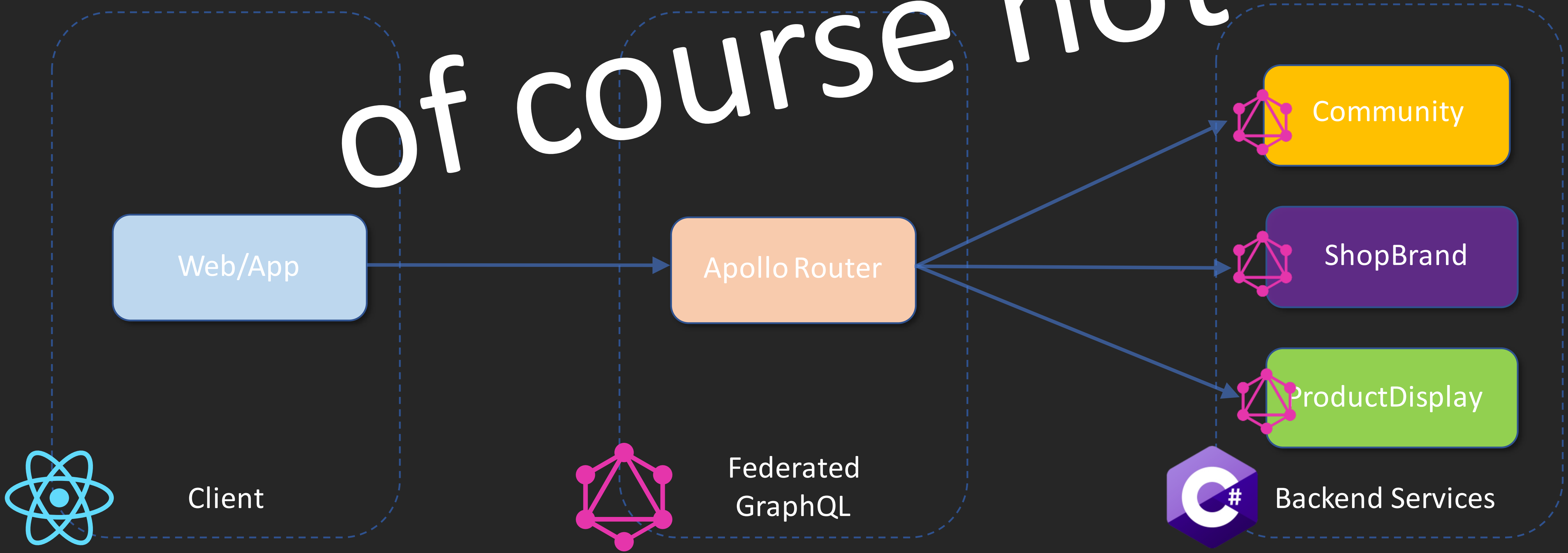
No proper graph

REST over GraphQL?

“Durchlauf erhitzer”

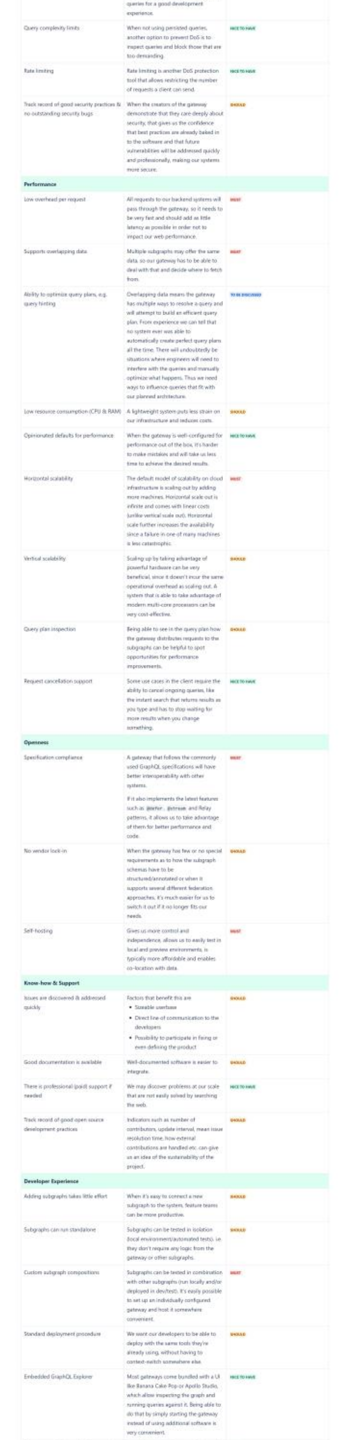
Salvation?

of course not

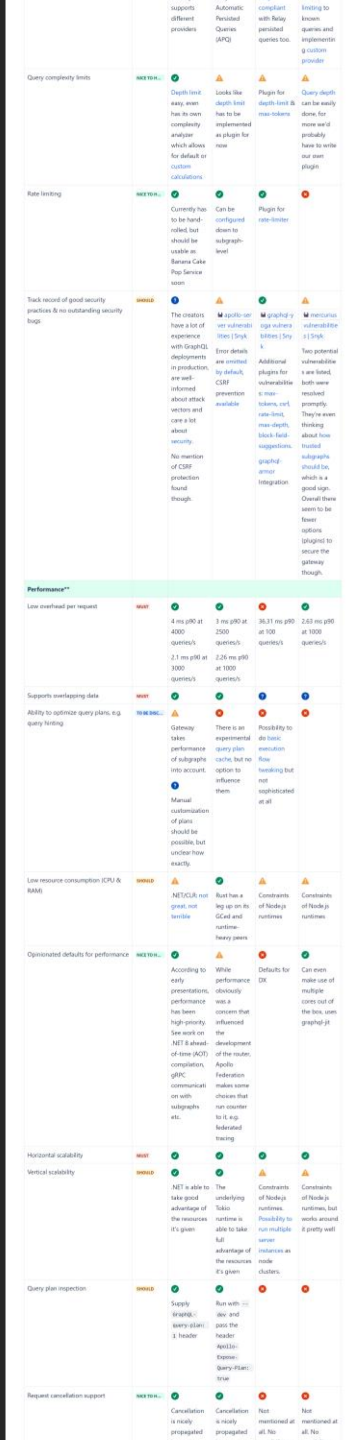


Gateway Evaluation

- 12 candidates: Apollo, Hasura, Mesh, Mercurius, Wundergraph, …
- Reduced to Fusion, Apollo Router, GraphQL Mesh and Mercurius
- Last two standing: Apollo and Fusion
- 🥇 Hot Chocolate Fusion 🥇



	queries for a good development experience.	
Query complexity limits	When not using persisted queries, another option to prevent DoS is to inspect queries and block those that are too demanding.	NICE TO HAVE
Rate limiting	Rate limiting is another DoS protection tool that allows restricting the number of requests a client can send.	NICE TO HAVE
Track record of good security practices & no outstanding security bugs	When the creators of the gateway demonstrate that they care deeply about security, that gives us the confidence that best practices are already baked in to the software and that future vulnerabilities will be addressed quickly and professionally, making our systems more secure.	SHOULD
Performance		
Low overhead per request	All requests to our backend systems will pass through the gateway, so it needs to be very fast and should add as little latency as possible in order not to impact our web performance.	MUST
Supports overlapping data	Multiple subgraphs may offer the same data, so our gateway has to be able to deal with that and decide where to fetch from.	MUST
Ability to optimize query plans, e.g. query hinting	Overlapping data means the gateway has multiple ways to resolve a query and will attempt to build an efficient query plan. From experience we can tell that no system ever was able to automatically create perfect query plans all the time. There will undoubtedly be situations where engineers will need to interfere with the queries and manually optimize what happens. Thus we need ways to influence queries that fit with our planned architecture.	TO BE DISCUSSED
Low resource consumption (CPU & RAM)	A lightweight system puts less strain on our infrastructure and reduces costs.	SHOULD
Opinionated defaults for performance	When the gateway is well-configured for performance out of the box, it's harder to make mistakes and will take us less time to achieve the desired results.	NICE TO HAVE
Horizontal scalability	The default model of scalability on cloud infrastructure is scaling out by adding more machines. Horizontal scale out is infinite and comes with linear costs (unlike vertical scale out). Horizontal scale further increases the availability since a failure in one of many machines is less catastrophic.	MUST
Vertical scalability	Scaling up by taking advantage of powerful hardware can be very beneficial, since it doesn't incur the same operational overhead as scaling out. A system that is able to take advantage of modern multi-core processors can be very cost-effective.	SHOULD
Query plan inspection	Being able to see in the query plan how the gateway distributes requests to the subgraphs can be helpful to spot opportunities for performance improvements.	SHOULD
Request cancellation support	Some use cases in the client require the ability to cancel ongoing queries, like the instant search that returns results as you type and has to stop waiting for more results when you change something.	NICE TO HAVE
Openness		
Specification compliance	A gateway that follows the commonly used GraphQL specifications will have better interoperability with other systems. If it also implements the latest features such as `@defer`, `@stream` and Relay patterns, it allows us to take advantage of them for better performance and code.	MUST
No vendor lock-in	When the gateway has few or no special requirements as to how the subgraph schemas have to be structured/annotated or when it supports several different federation approaches, it's much easier for us to switch it out if it no longer fits our needs.	SHOULD
Self-hosting	Gives us more control and independence, allows us to easily test in local and preview environments, is typically more affordable and enables co-location with data.	MUST
Know-how & Support		
Issues are discovered & addressed quickly	Factors that benefit this are • Sizeable userbase • Direct line of communication to the developers • Possibility to participate in fixing or even defining the product	SHOULD
Good documentation is available	Well-documented software is easier to integrate.	SHOULD
There is professional (paid) support if needed	We may discover problems at our scale that are not easily solved by searching the web.	NICE TO HAVE
Track record of good open source development practices	Indicators such as number of contributors, update interval, mean issue resolution time, how external contributions are handled etc. can give us an idea of the sustainability of the project.	SHOULD
Developer Experience		
Adding subgraphs takes little effort	When it's easy to connect a new subgraph to the system, feature teams can be more productive.	SHOULD
Subgraphs can run standalone	Subgraphs can be tested in isolation (local environment/automated tests), i.e. they don't require any logic from the gateway or other subgraphs.	SHOULD
Custom subgraph compositions	Subgraphs can be tested in combination with other subgraphs (run locally and/or deployed in dev/test). It's easily possible to set up an individually configured gateway and host it somewhere convenient.	MUST
Standard deployment procedure	We want our developers to be able to deploy with the same tools they're already using, without having to context-switch somewhere else.	SHOULD
Embedded GraphQL Explorer	Most gateways come bundled with a UI like Banana Cake Pop or Apollo Studio, which allow inspecting the graph and running queries against it. Being able to do that by simply starting the gateway instead of using additional software is very convenient.	NICE TO HAVE

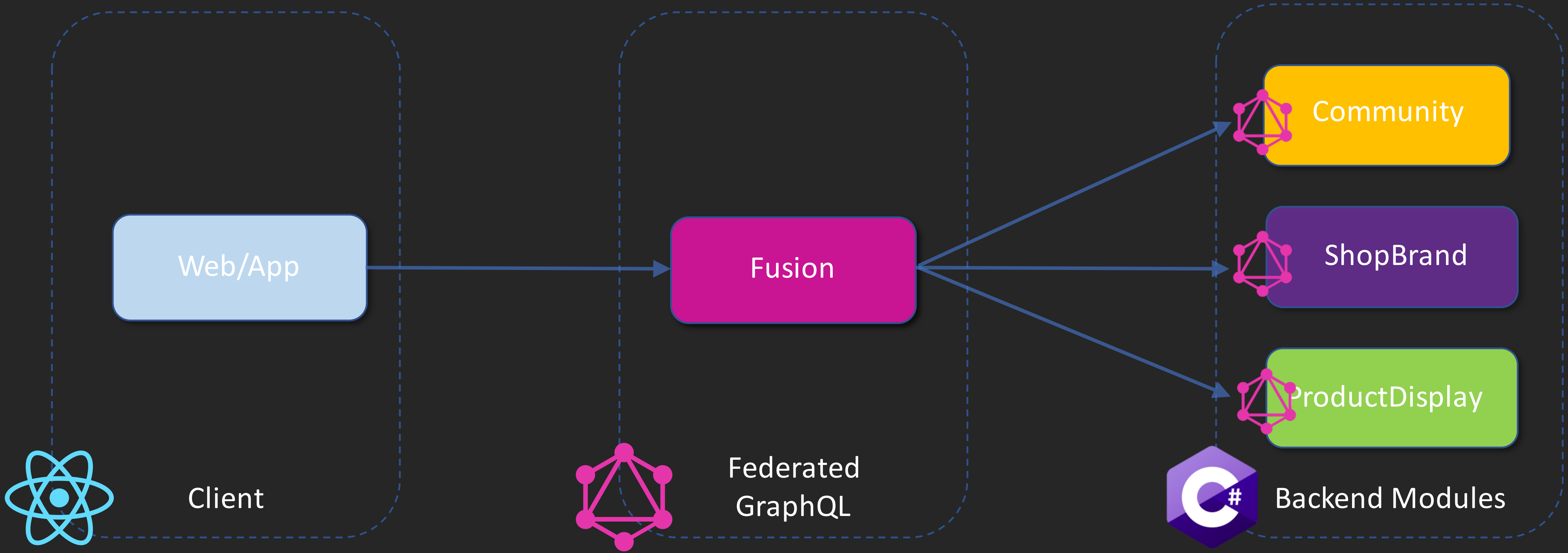


		supports different providers	Automatic Persisted Queries (APQ)	compliant with Relay persisted queries too.	limiting to known queries and implementing custom provider
Query complexity limits	NICE TO H...	✓ Depth limit easy, even has its own complexity analyzer which allows for default or custom calculations	⚠ Looks like depth limit has to be implemented as plugin for now	⚠ Plugin for depth-limit & max-tokens	⚠ Query depth can be easily done, for more we'd probably have to write our own plugin
Rate limiting	NICE TO H...	✓ Currently has to be hand-rolled, but should be usable as Banana Cake Pop Service soon	✓ Can be configured down to subgraph-level	✓ Plugin for rate-limiter	✗
Track record of good security practices & no outstanding security bugs	SHOULD	? The creators have a lot of experience with GraphQL deployments in production, are well-informed about attack vectors and care a lot about security. No mention of CSRF protection found though.	⚠ apollo-server vulnerabilities | Snyk Error details are omitted by default, CSRF prevention available	✓ graphql-yoga vulnerabilities | Snyk Additional plugins for vulnerabilities: max-tokens, csrf, rate-limit, max-depth, block-field-suggestions. graphql-armor Integration.	⚠ mercurius vulnerabilities | Snyk Two potential vulnerabilities are listed, both were resolved promptly. They're even thinking about how trusted subgraphs should be, which is a good sign. Overall there seem to be fewer options (plugins) to secure the gateway though.
Performance**					
Low overhead per request	MUST	✓ 4 ms p90 at 4000 queries/s 2.1 ms p90 at 3000 queries/s	✓ 3 ms p90 at 2500 queries/s 2.26 ms p90 at 1000 queries/s	✗ 36.31 ms p90 at 100 queries/s	✓ 2.63 ms p90 at 1000 queries/s
Supports overlapping data	MUST	✓	✓	?	?
Ability to optimize query plans, e.g. query hinting	TO BE DISC...	⚠ Gateway takes performance of subgraphs into account. ? Manual customization of plans should be possible, but unclear how exactly.	✗ There is an experimental query plan cache, but no option to influence them	✗ Possibility to do basic execution flow tweaking but not sophisticated at all	✗
Low resource consumption (CPU & RAM)	SHOULD	⚠ .NET/CLR: not great, not terrible	✓ Rust has a leg up on its GCed and runtime-heavy peers	⚠ Constraints of Node.js runtimes	⚠ Constraints of Node.js runtimes
Opinionated defaults for performance	NICE TO H...	✓ According to early presentations, performance has been high-priority. See work on .NET 8 ahead-of-time (AOT) compilation, gRPC communication with subgraphs etc.	⚠ While performance obviously was a concern that influenced the development of the router, Apollo Federation makes some choices that run counter to it, e.g. federated tracing	✗ Defaults for DX	✓ Can even make use of multiple cores out of the box, uses graphql-jit
Horizontal scalability	MUST	✓	✓	✓	✓
Vertical scalability	SHOULD	✓ .NET is able to take good advantage of the resources it's given	✓ The underlying Tokio runtime is able to take full advantage of the resources it's given	⚠ Constraints of Node.js runtimes. Possibility to run multiple server instances as node clusters.	⚠ Constraints of Node.js runtimes, but works around it pretty well
Query plan inspection	SHOULD	✓ Supply `GraphQL-Query-Plan: 1` header	✓ Run with `--dev` and pass the header `Apollo-Expose-Query-Plan: true`	✗	✗
Request cancellation support	NICE TO H...	✓ Cancellation is nicely propagated	✓ Cancellation is nicely propagated	✗ Not mentioned at all. No	✗ Not mentioned at all. No

Gateway Deployment

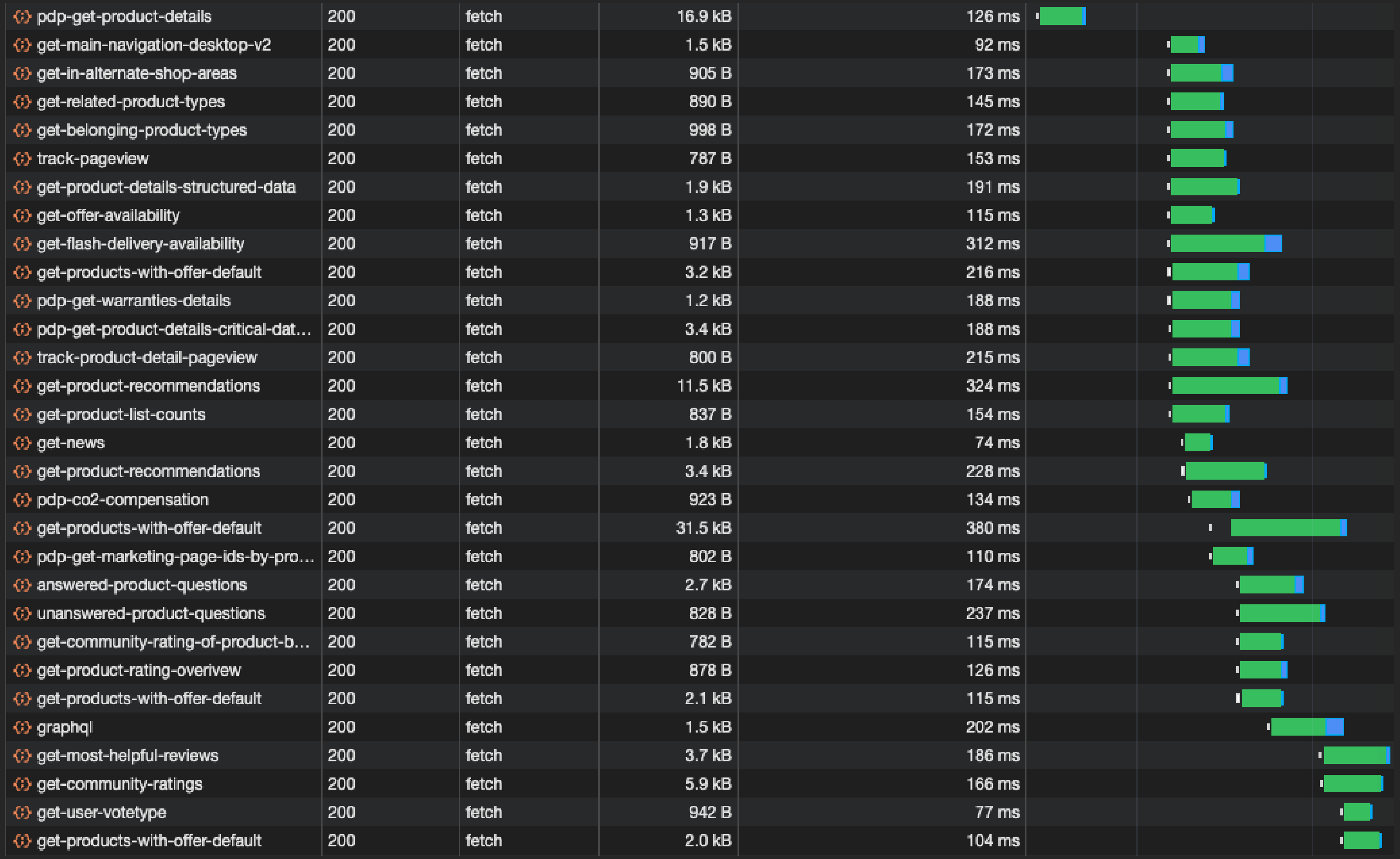
- Deploying the gateway is straightforward
- Cheap DIY solution: Redis + some pipelines
- In production since August
- Challenges
  - Bleeding edge gateway
  - Evolving spec
  - Schema registry

What Changed?



Meanwhile in the FrontEnd

- Next.js application with pages Router
- Apollo Client as our current GraphQL client
- Queries per component and **not** per page

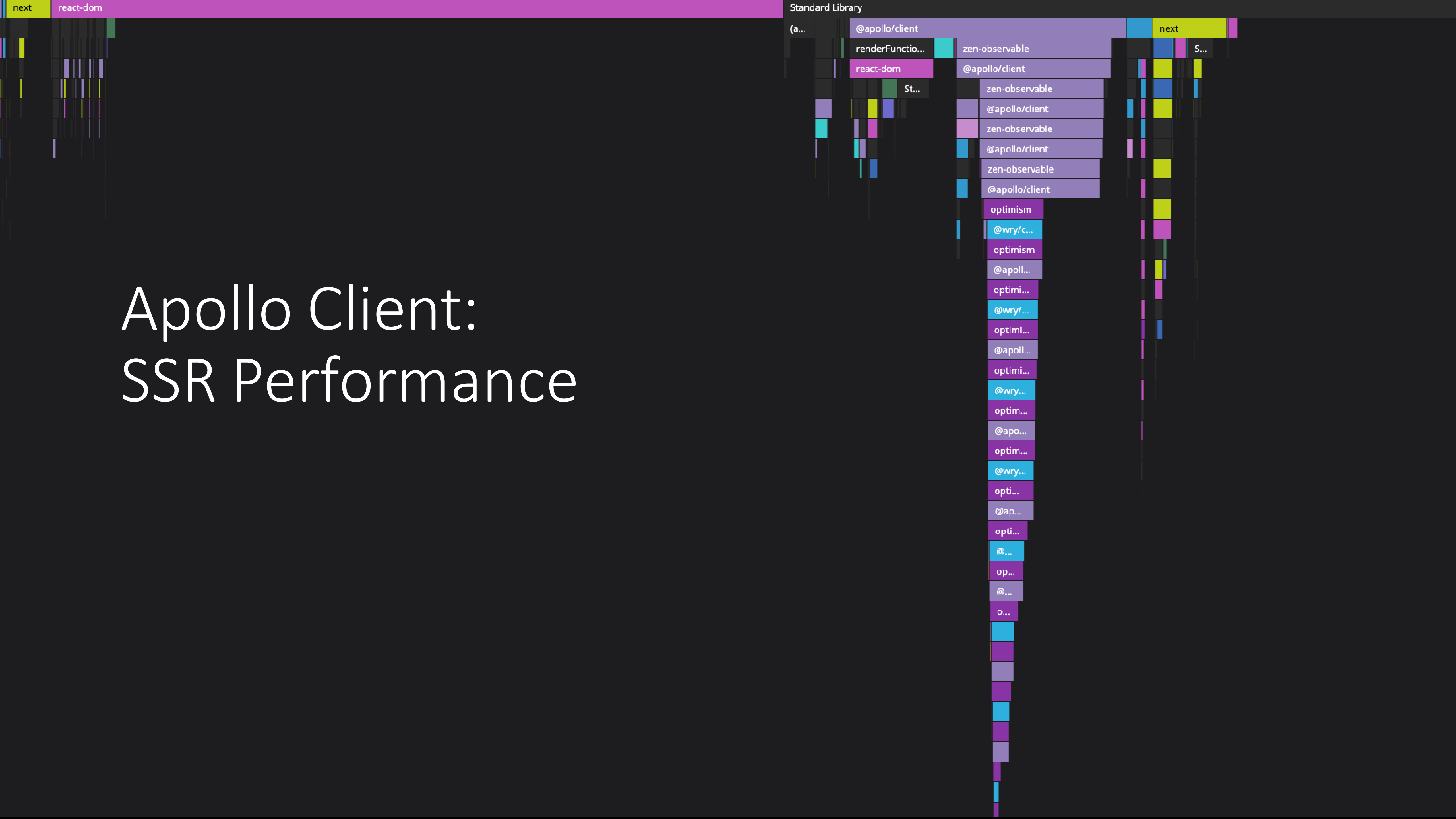


pdp-get-product-details	200	fetch	16.9 kB	126 ms
get-main-navigation-desktop-v2	200	fetch	1.5 kB	92 ms
get-in-alternate-shop-areas	200	fetch	905 B	173 ms
get-related-product-types	200	fetch	890 B	145 ms
get-belonging-product-types	200	fetch	998 B	172 ms
track-pageview	200	fetch	787 B	153 ms
get-product-details-structured-data	200	fetch	1.9 kB	191 ms
get-offer-availability	200	fetch	1.3 kB	115 ms
get-flash-delivery-availability	200	fetch	917 B	312 ms
get-products-with-offer-default	200	fetch	3.2 kB	216 ms
pdp-get-warranties-details	200	fetch	1.2 kB	188 ms
pdp-get-product-details-critical-dat…	200	fetch	3.4 kB	188 ms
track-product-detail-pageview	200	fetch	800 B	215 ms
get-product-recommendations	200	fetch	11.5 kB	324 ms
get-product-list-counts	200	fetch	837 B	154 ms
get-news	200	fetch	1.8 kB	74 ms
get-product-recommendations	200	fetch	3.4 kB	228 ms
pdp-co2-compensation	200	fetch	923 B	134 ms
get-products-with-offer-default	200	fetch	31.5 kB	380 ms
pdp-get-marketing-page-ids-by-pro…	200	fetch	802 B	110 ms
answered-product-questions	200	fetch	2.7 kB	174 ms
unanswered-product-questions	200	fetch	828 B	237 ms
get-community-rating-of-product-b…	200	fetch	782 B	115 ms
get-product-rating-overivew	200	fetch	878 B	126 ms
get-products-with-offer-default	200	fetch	2.1 kB	115 ms
graphql	200	fetch	1.5 kB	202 ms
get-most-helpful-reviews	200	fetch	3.7 kB	186 ms
get-community-ratings	200	fetch	5.9 kB	166 ms
get-user-votetype	200	fetch	942 B	77 ms
get-products-with-offer-default	200	fetch	2.0 kB	104 ms

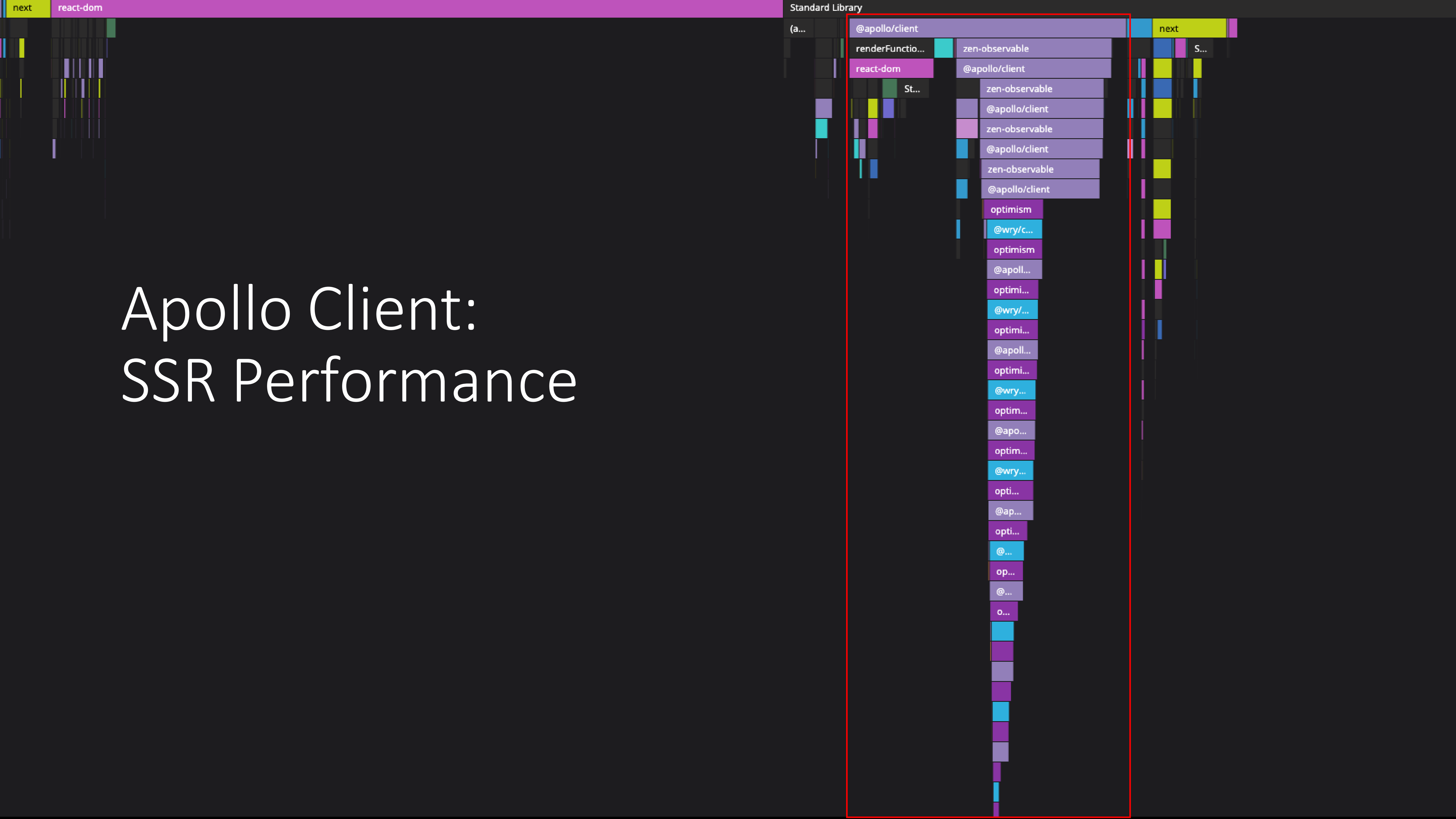
Server-side rendering with Apollo Client

- getDataFromTree re-renders page until all nested queries are finished
- At least one additional full render per page
- Queries at two levels lead to **four** re-renders of the page

Apollo Client: SSR Performance

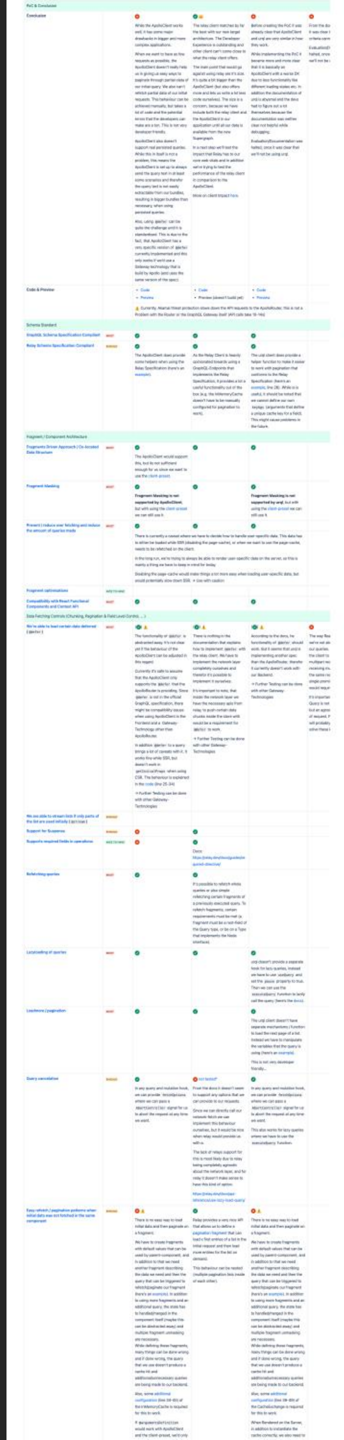


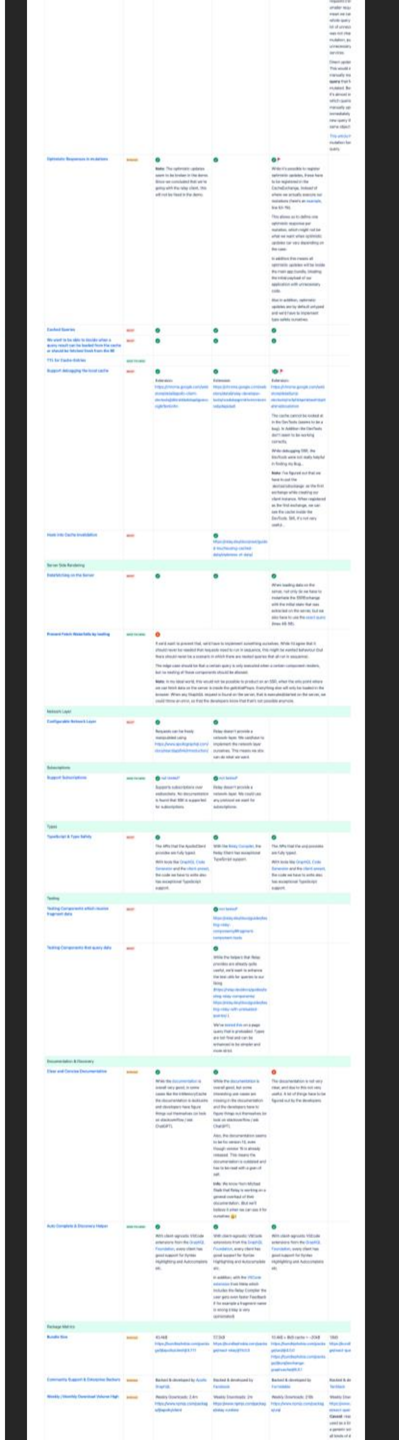
Apollo Client: SSR Performance



Yet Another Evaluation

- Comparing GraphQL clients like Apollo, urql, etc.
- 👑 Relay





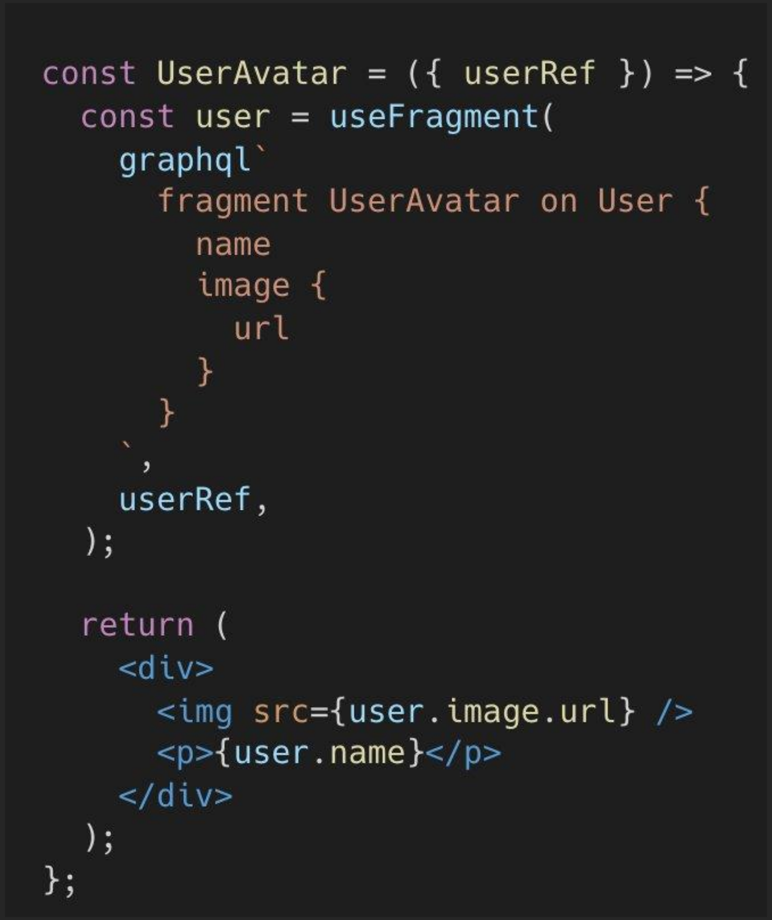
Robert Balicki

he/him

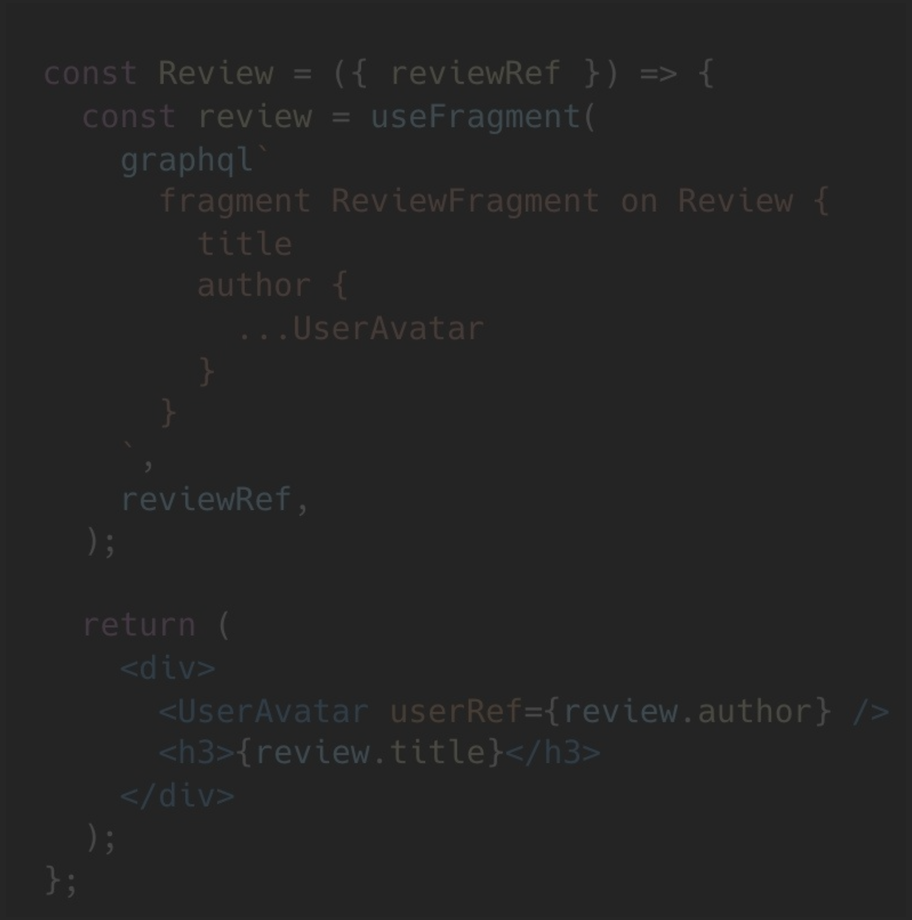
Re-introducing Relay

https://www.youtube.com/watch?v=lhVGdErZuN4

Relay in a 🥜-shell

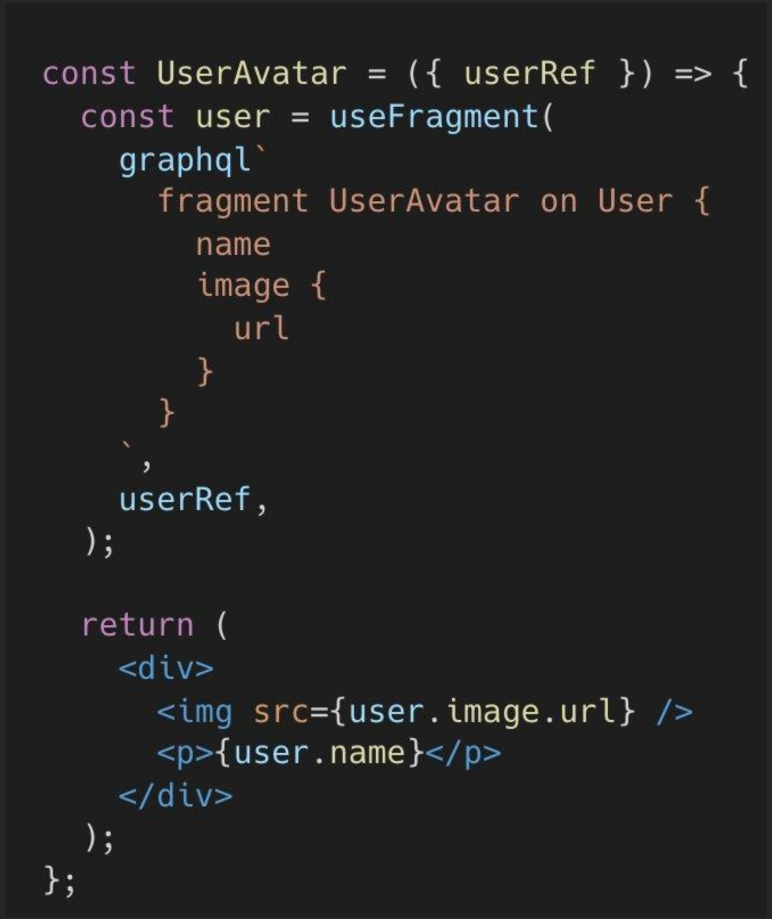


```
const UserAvatar = ({ userRef }) => {
  const user = useFragment(
    graphql`
      fragment UserAvatar on User {
        name
        image {
          url
        }
      }
    `,
    userRef,
  );

  return (
    <div>
      <img src={user.image.url} />
      <p>{user.name}</p>
    </div>
  );
};
```

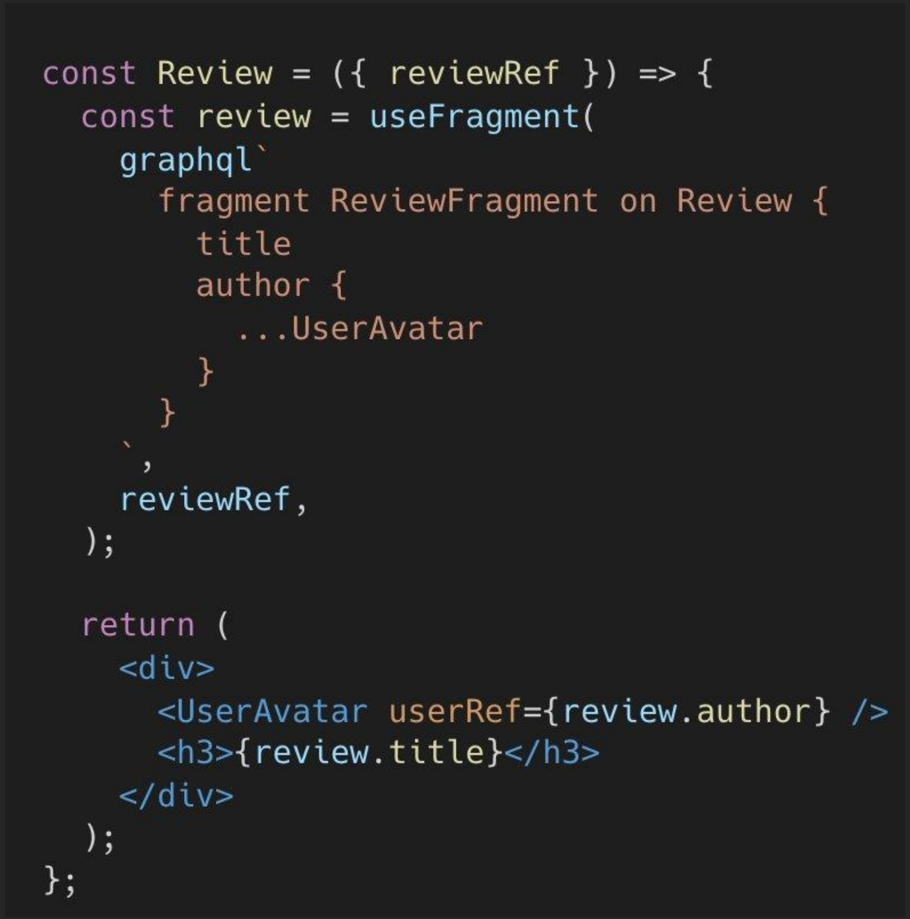


```
const Review = ({ reviewRef }) => {
  const review = useFragment(
    graphql`
      fragment ReviewFragment on Review {
        title
        author {
          ...UserAvatar
        }
      }
    `,
    reviewRef,
  );

  return (
    <div>
      <UserAvatar userRef={review.author} />
      <h3>{review.title}</h3>
    </div>
  );
};
```

Relay in a 🥜-shell



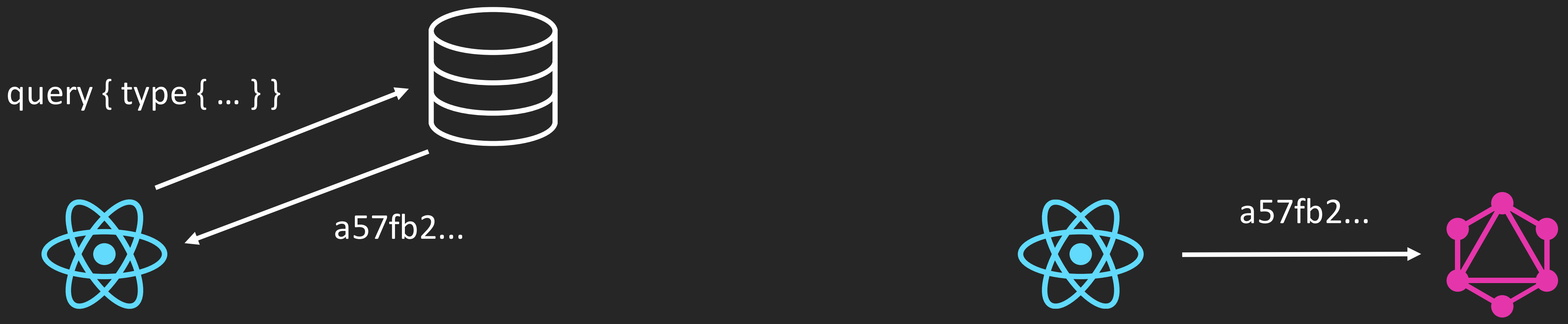
```
const UserAvatar = ({ userRef }) => {
  const user = useFragment(
    graphql`
      fragment UserAvatar on User {
        name
        image {
          url
        }
      }
    `,
    userRef,
  );

  return (
    <div>
      <img src={user.image.url} />
      <p>{user.name}</p>
    </div>
  );
};
```



```
const Review = ({ reviewRef }) => {
  const review = useFragment(
    graphql`
      fragment ReviewFragment on Review {
        title
        author {
          ...UserAvatar
        }
      }
    `,
    reviewRef,
  );

  return (
    <div>
      <UserAvatar userRef={review.author} />
      <h3>{review.title}</h3>
    </div>
  );
};
```

Setting up Relay

- Relatively easy, but sometimes requires looking "under the hood"
- Some headaches with SSR
- "Persisted Operations" for free 🤑

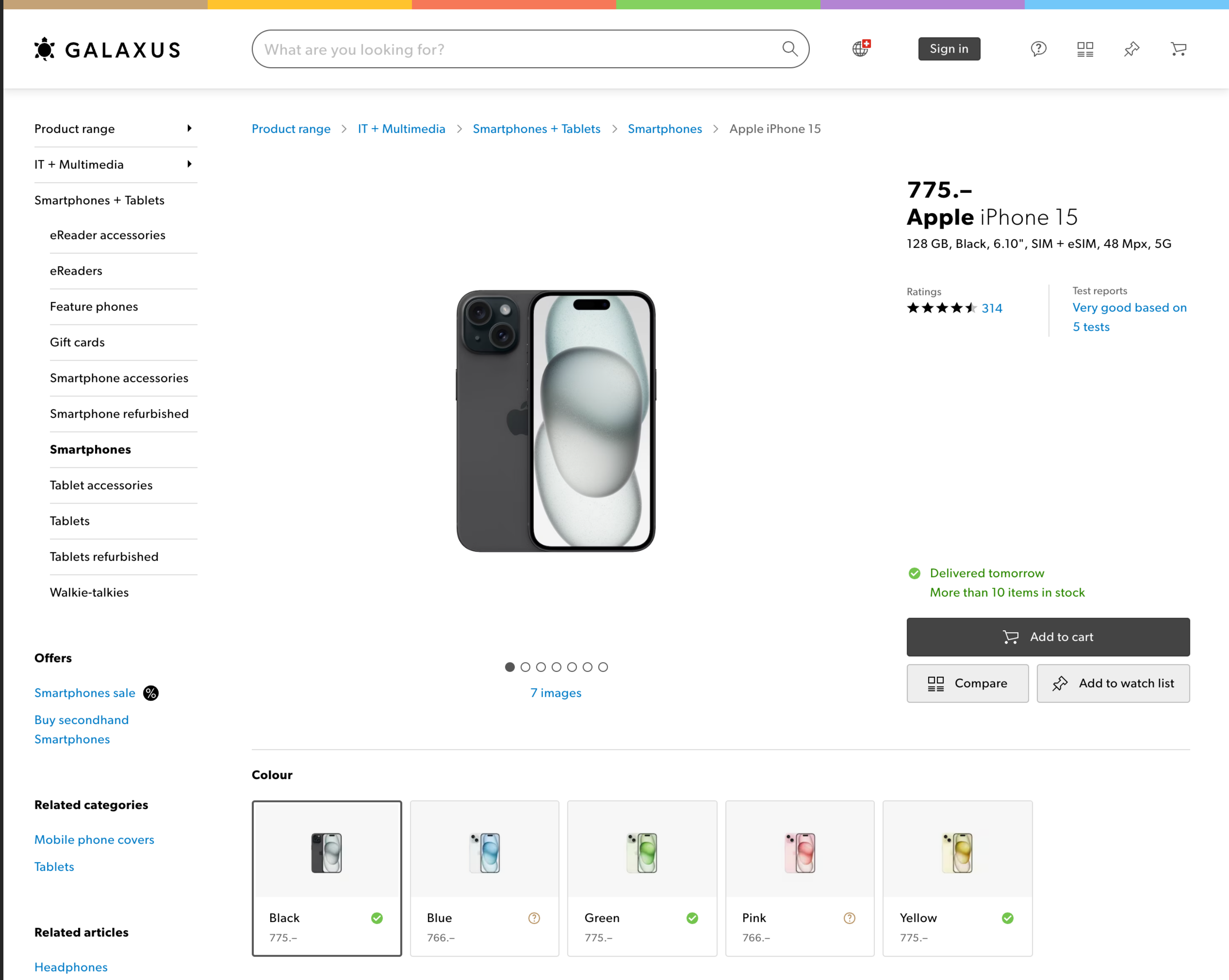


query { type { … } }

a57fb2...

a57fb2...

AB-Test 🎯



Product range > IT + Multimedia > Smartphones + Tablets > Smartphones > Apple iPhone 15

- Product range
- IT + Multimedia
- Smartphones + Tablets
  - eReader accessories
  - eReaders
  - Feature phones
  - Gift cards
  - Smartphone accessories
  - Smartphone refurbished
  - **Smartphones**
  - Tablet accessories
  - Tablets
  - Tablets refurbished
  - Walkie-talkies

Offers

Smartphones sale %

Buy secondhand Smartphones

Related categories

Mobile phone covers

Tablets

Related articles

Headphones

775.–

Apple iPhone 15

128 GB, Black, 6.10", SIM + eSIM, 48 Mpx, 5G

Ratings ★★★★★ 314

Test reports Very good based on 5 tests

✓ Delivered tomorrow
More than 10 items in stock

Colour

Black ✓	775.–
Blue ?	766.–
Green ✓	775.–
Pink ?	766.–
Yellow ✓	775.–

AB-Test: Results

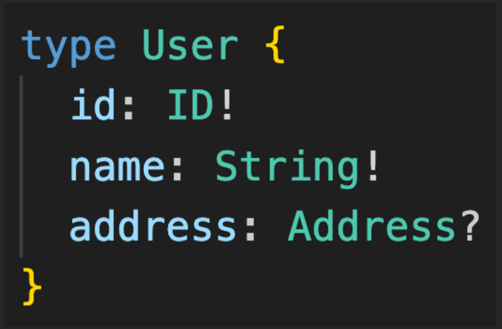
Metric	**Change (P75)**
Front-End Latency	52% reduction
Time To First Byte (TTFB)	18% reduction
Largest Contentful Paint (LCP)	9% reduction

Next steps

- Soft Rollout for feature teams
- Onboarding workshops
- Migration of existing pages starting in Q2

Developments we are excited about

- True Schema Nullability and no more error bubbling



```
type User {
  id: ID!
  name: String!
  address: Address?
}
```

- @defer support in the Fusion Gateway
- Banana Cake Pop (Schema & Client Registry)

We're hiring 🥳

Information about the talk

https://t.ly/yXhiu

https://t.ly/Sy7fM

Information about the talk

https://t.ly/yXhiu

We're hiring 🥳

https://t.ly/Sy7fM