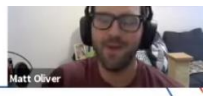
TRACK
Architectures You've Always Wondered About

SESSION

Federated GraphQL to Solve Service Sprawl at Major League Baseball

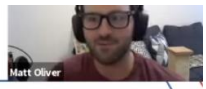
Olessya Medvedeva and Matt Oliver discuss how they have begun to implement a Federated GraphQL architecture to solve the issue of service discovery, sprawl and ultimately getting the data needed.

Who are we? What is Web Platform?

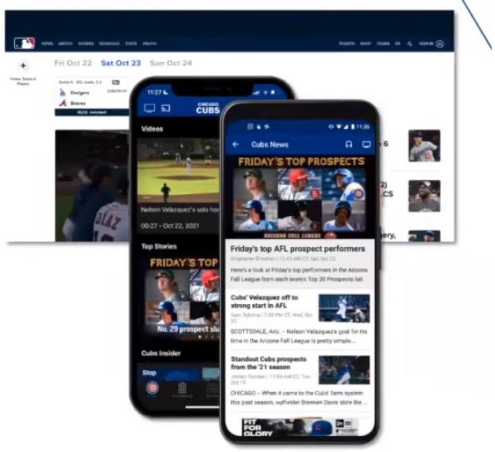


- Matt Oliver - Senior Engineering Manager
- Olessya Medvedeva - Software Engineer
- Web Platform handles all base architecture, infra and DevOps.
  - Supports multiple frontend teams, 30+ developers
  - Forward thinking, leading architecture vision.
- Responsibilities include the Server Side Rendering architecture (React/Next) and supporting services such as personalization.
  - **Challenging because serving fully rendered pages takes coordinating data from many disparate services.**

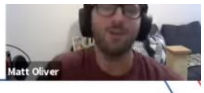
The Problem



"We have different clients needing different data from many disparate systems maintained by different teams."

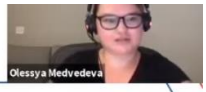


Specifics



- Low visibility into who is making calls
  - Unknown clients make model changes hard
- Evidence of redundant calls but system too complex to determine where
  - Prone to DDOSing ourselves
- Burned by third party integrations, isolate clients from backend churn
- Disparate caching causing confusion
- Upstream failures handled poorly

Client/Server Communication Today

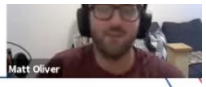


REST services where the client does the heavy lifting

What's wrong with this?

- Clients are chatty
- Response payloads all-or-nothing
- Backend data model tied to frontend implementation
- Backend service exposure

Client/Server Communication Today



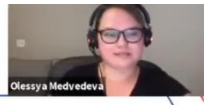
Mashups simplifying complex data requirements

What's wrong with this?

- Increased complexity
- Maintenance of mashups
- Duplication of data access
- No holistic view of api surface

Decided to explore GraphQL

What is GraphQL?



- **Graph Q**uery **L**anguage
- Developed by Facebook in 2012, first public release in 2015.
- Using the query language, request the data you want from the server.

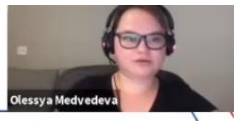
POST /graphql

```
query {
  getTeam(ids: [ 147 ]) {
    name
  }
}
```

→

```
{
  "data": {
    "getTeams": [{
      "name": "New York Yankees"
    }]
  }
}
```

Anatomy Of A GQL Service



Models

Describes request and response schema

```
type Query {
  getTeams(ids: [Int]): [Team]
}

type Team {
  id: Int
  name: String
  ...
}
```

Resolvers

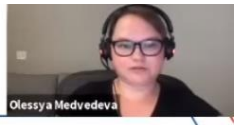
Translates request into response

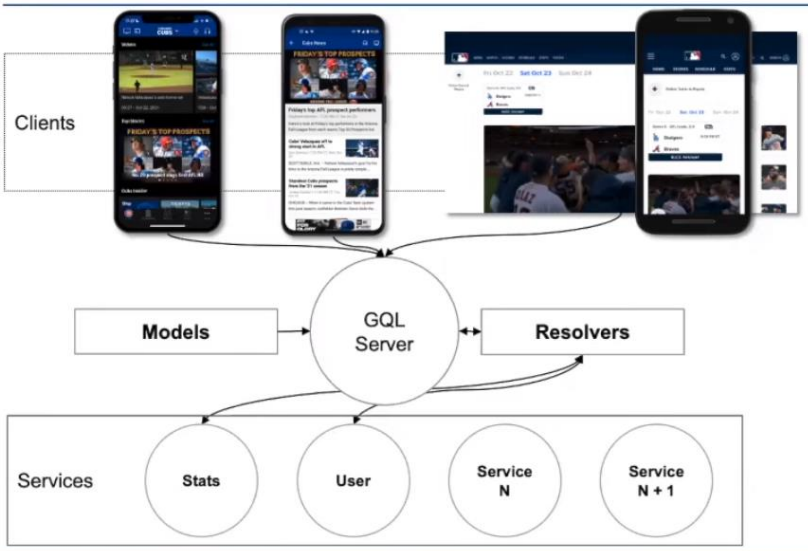
```
module.exports = {
  getTeams: (_, { ids }, { stats }) => {
    return stats
      .getTeams(ids)
      .then(({
        teamId,
        teamName
      })) => ({
        id: teamId,
        name: teamName,
      });
  }
```

Services

Provide internal data to resolvers

Anatomy Of A GQL Service





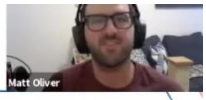
Pros

- Code all in one place, single source of truth
- Easier deployment
- Updates to models clear

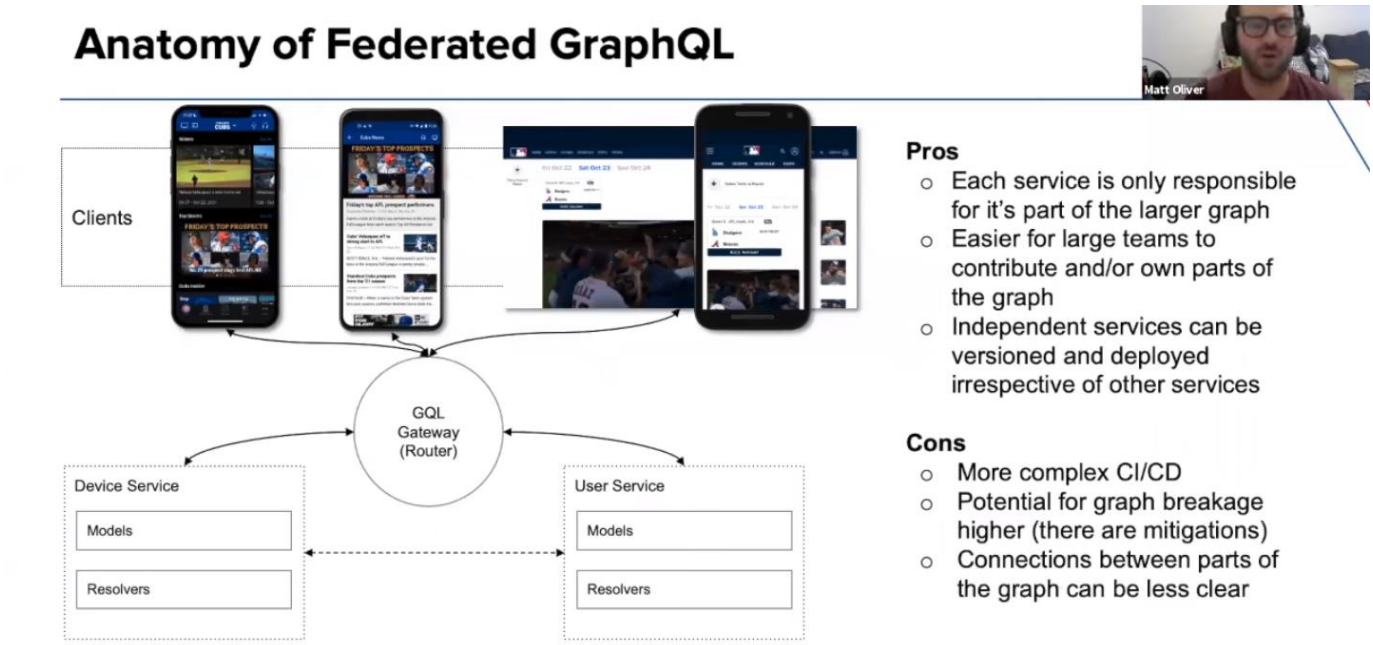
Cons

- Contributing with large teams cumbersome
- Models are tightly coupled
- Single point of failure

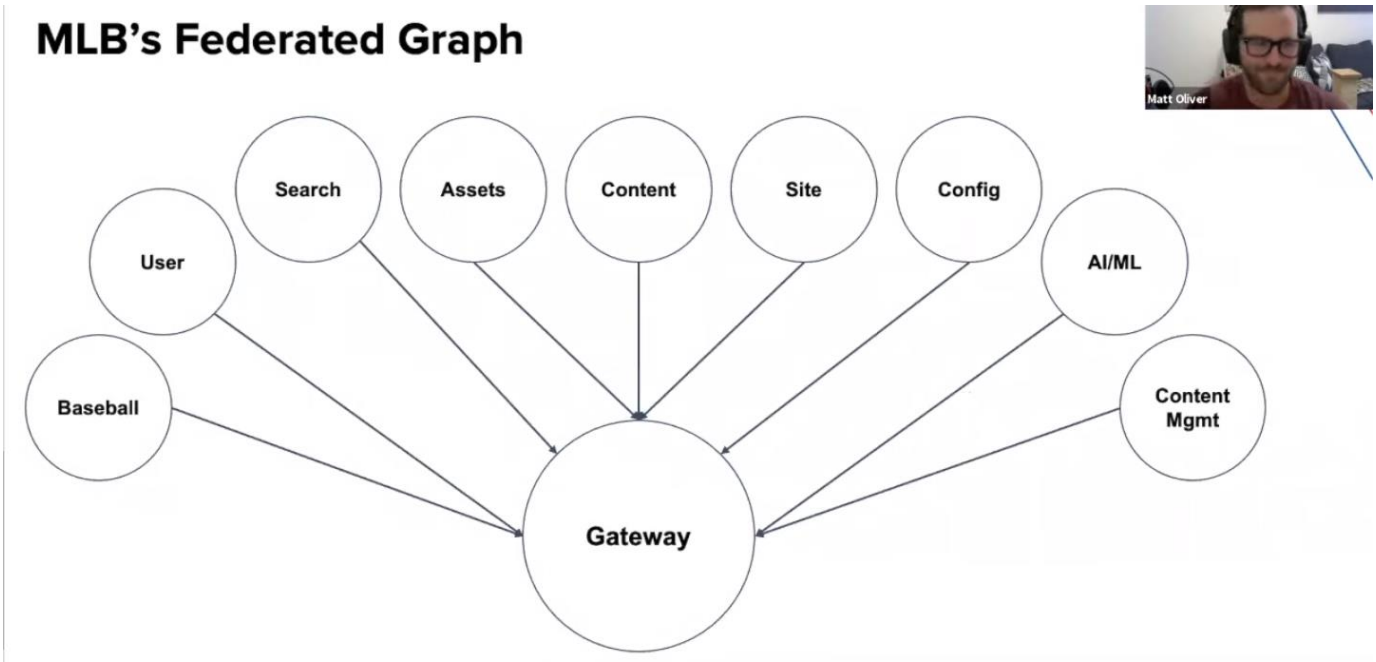
Enter APOLLO



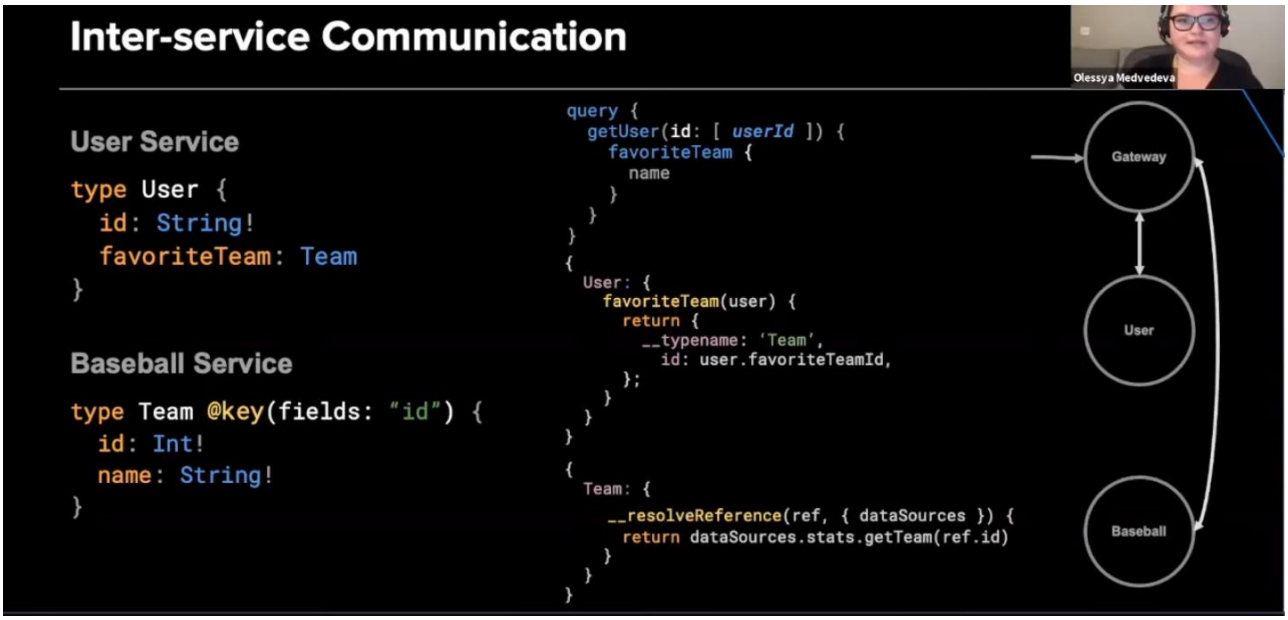
- Private company formed around GraphQL
- Extended the GraphQL specification, specifically around Federation
- Federation is a topology where you have multiple independent services, unified by one “gateway”

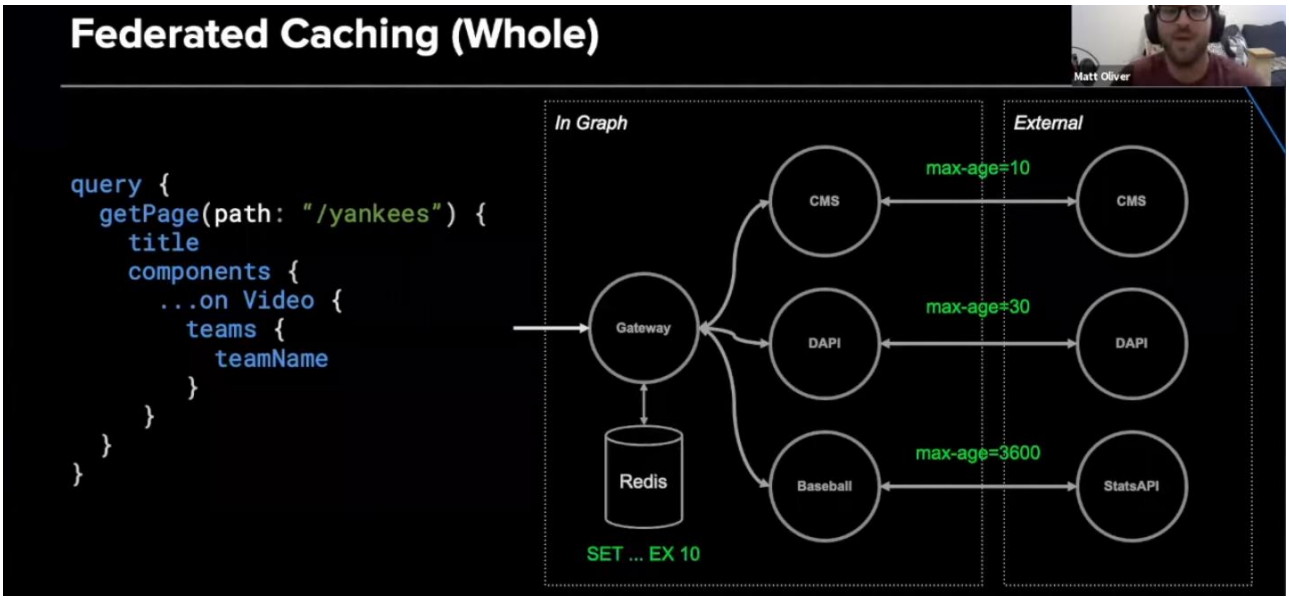


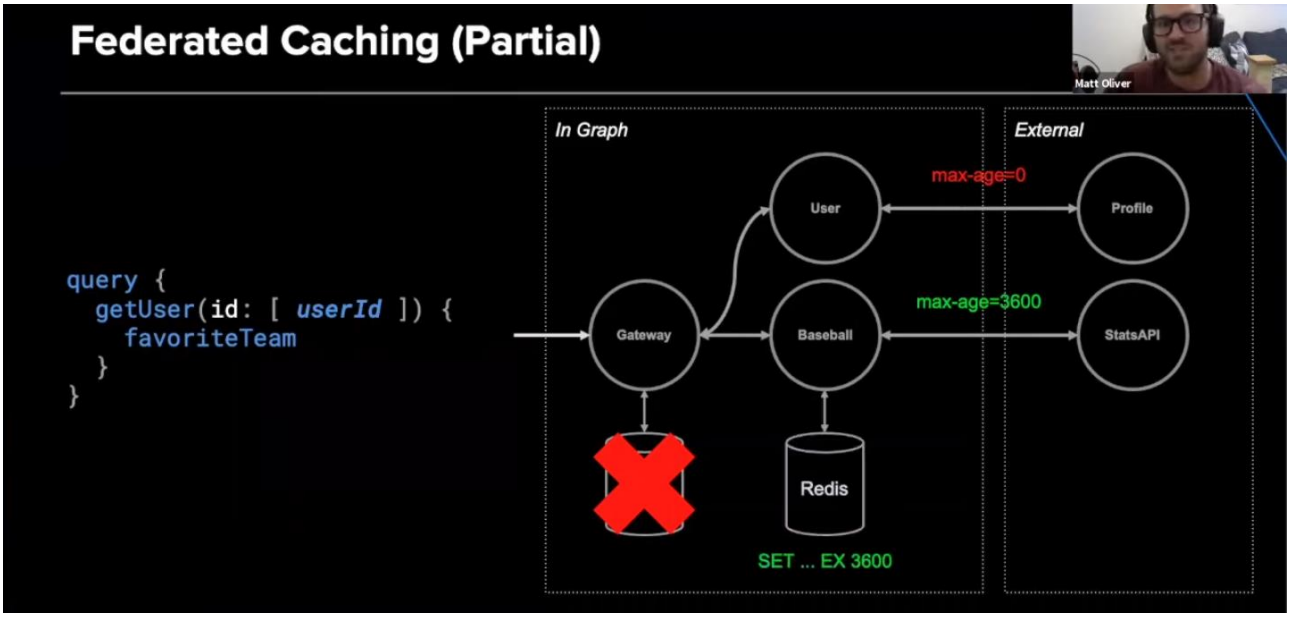
We now have subgraphs and can now do interservice communication between the different sub-graphs via querying



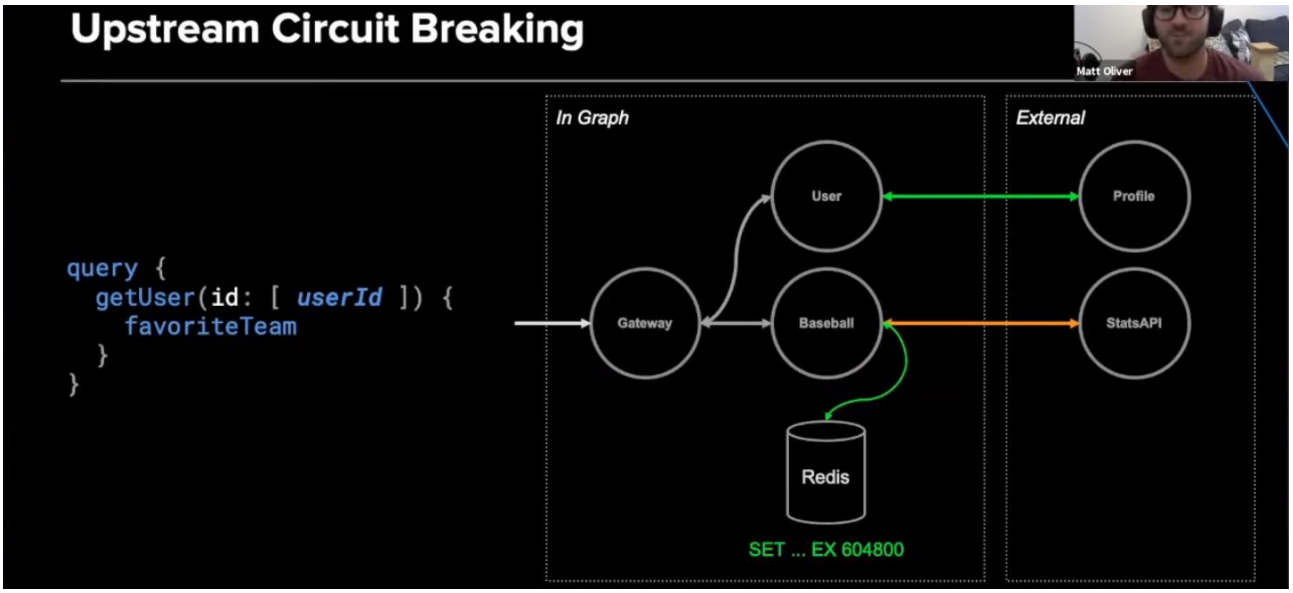
Developers can extend existing models within their own models

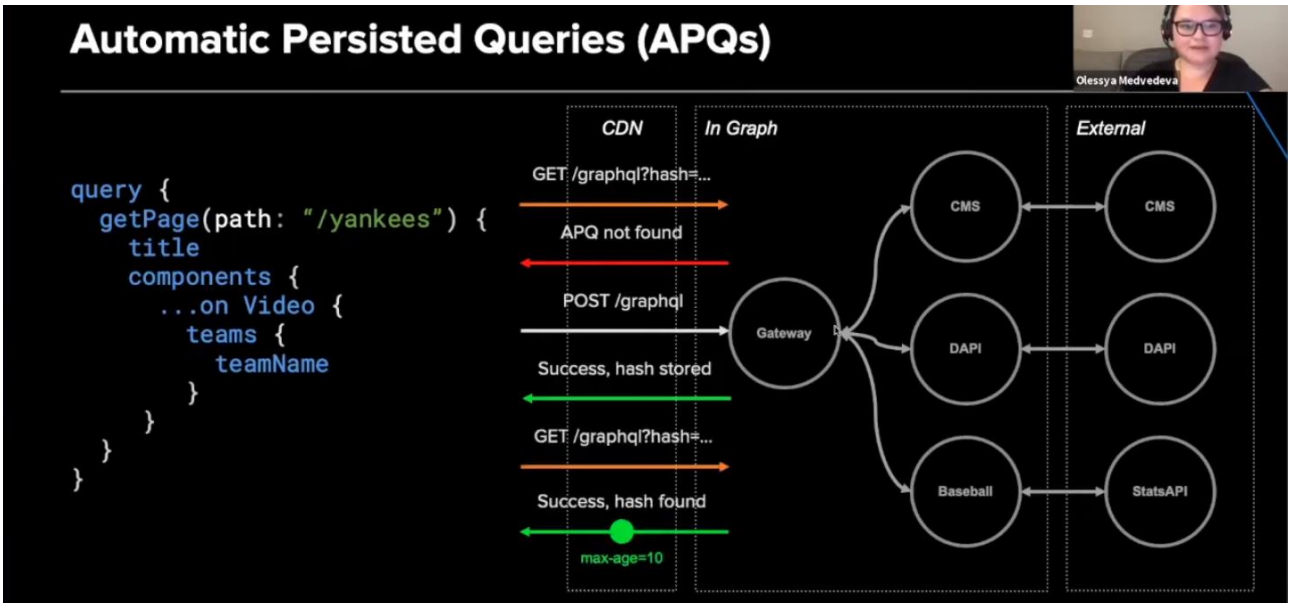




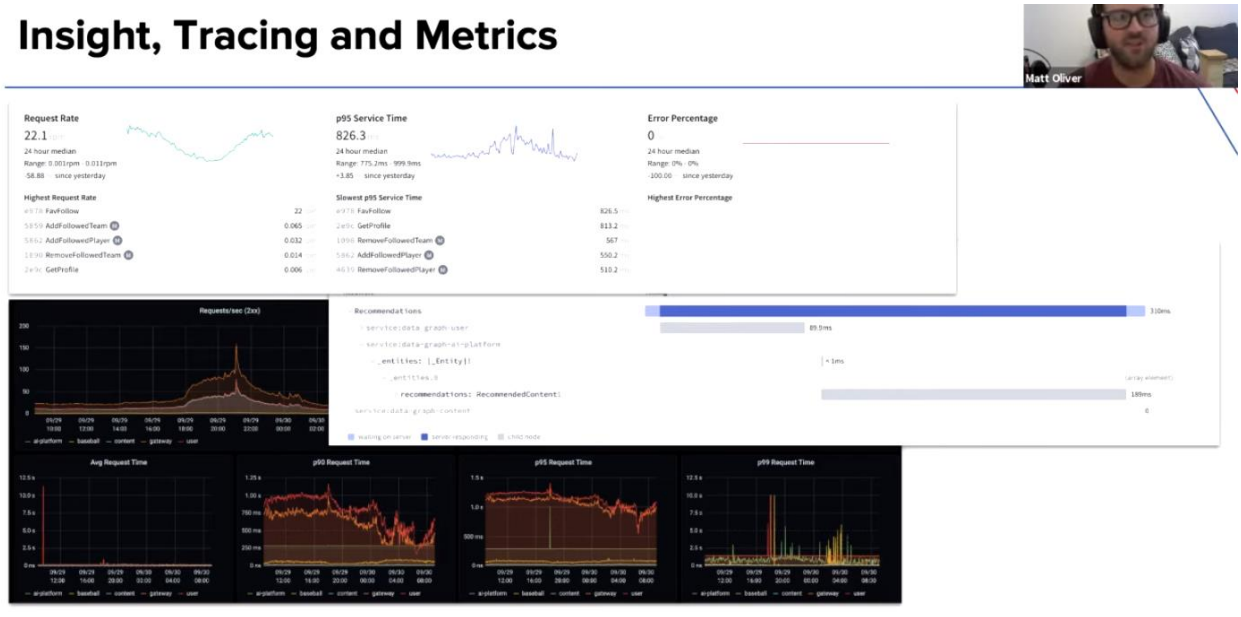


We can cache at the local service level and not at the Gateway level

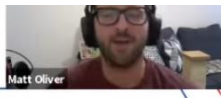




We can use APQs to cache and hash at the edge for better performance



Lessons Learned



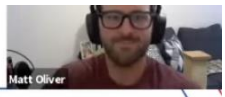
Wins

- Went into Production in May
- Subset of total services (60rqps at peak)
- Decreased number of calls between services
  - No more stitching within clients, payloads decreased
  - Teams leveraging federation, no longer need to worry about data access
- Increased visibility upstream with tracing
- Centralized data access reduces ambiguity and eases discovery

Caveats

- Upfront cost in learning GQL syntax/grammar
- Implicit federated communication learning curve
- Organizational buy-in a process
- Governance requires almost constant oversight and collaboration

Resources



- GraphQL Docs
  - https://graphql.org/learn/
- Apollo Docs
  - https://www.apollographql.com/docs/
- Apollo Federation
  - https://www.apollographql.com/docs/federation/
- Apollo GraphQL Summit, November 10 - 11
- Connect with Us!
  - Check out talk profile for LinkedIn links