lifely

GraphQL performance and security.

06 november 2019

Bryan te Beek
Lead Backend / Ops

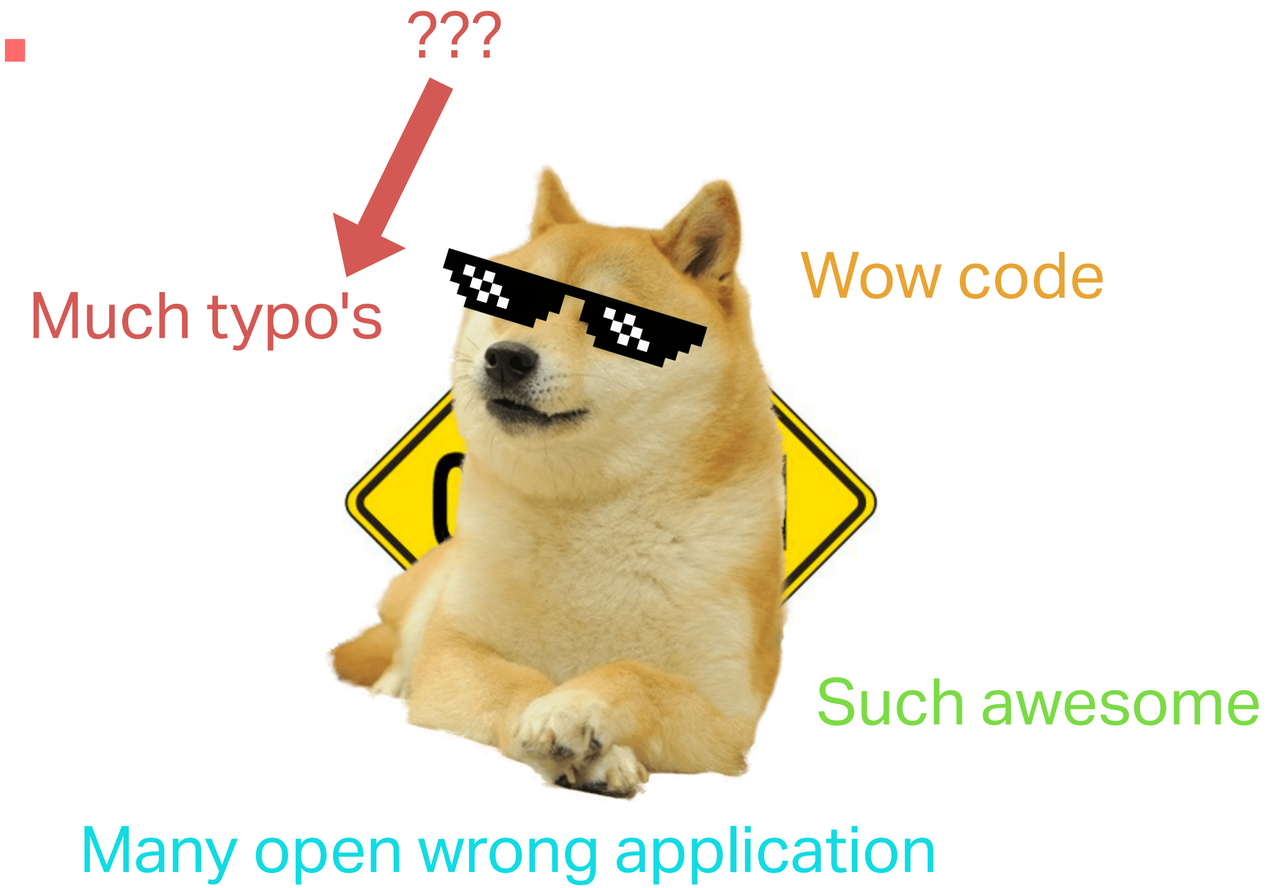
bryan@lifely.nl

@bryantebeek

GraphQL Performance and Security.

A fair warning...

A lot of demo's.



Performance in GraphQL.

- The N+1 problem
- Persisted queries

N+1 problem.

```
# Select all the people (1)
SELECT * FROM People

# Iterate over each person and get his friends
SELECT * FROM Friendship WHERE personId = :personId
```

- One select for all people
- For each person we run a query to get their friendships
- Really easy to hit this problem when using GraphQL
- ORM's make this really easy to do wrong

Demo 1: The N+1 problem.

N+1 problem.

- Easily solved using DataLoaders
- Create DataLoaders on context so they are per request! Otherwise your cache leaks to other user sessions.
- https://github.com/graphql/dataloader

Demo 2: DataLoaders.

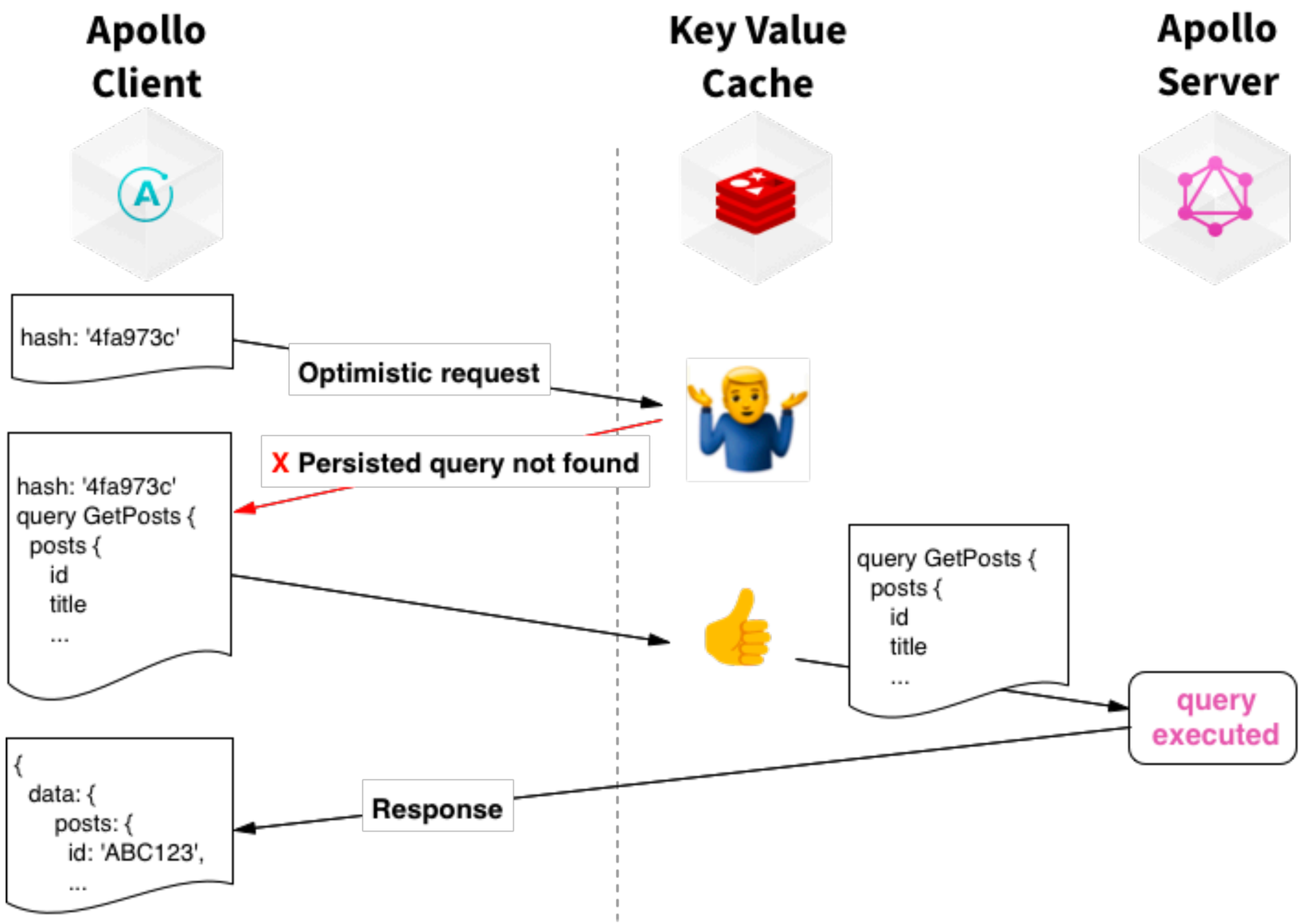
N+1 problem.

- Over-fetching of the Person when no fields are requested
- Object-centric vs field-centric data loading
- https://github.com/nick-keller/graphql-proxy

Demo 3: DataLoaders (field-centric).

Persisted queries.

- Allows caching of GET requests, just like in REST
- Easy way to allow query whitelisting (hash all frontend queries)
- https://www.apollographql.com/docs/apollo-server/performance/apq


Apollo Client
Key Value Cache
Apollo Server
hash: '4fa973c'
Optimistic request
X Persisted query not found
hash: '4fa973c'
query GetPosts {
posts {
id
title
...
query GetPosts {
posts {
id
title
...
query executed
{
data: {
posts: {
id: 'ABC123',
...
Response

```
// http://localhost:4000/graphql?
extensions=%7B%0A%20%20%20%20%22persistedQuery%22%3A%20%7B%0A%20%20%20%20%20%20%20%20%22versi
on%22%3A1,%0A%20%20%20%20%20%20%20%20%22sha256Hash%22%3A%2245ec52f8be059c86962446872ae55974c5
a27944d9e530358a79ae19e828f549%22%0A%20%20%20%20%7D%0A%7D


// extensions=
{
    "persistedQuery": {
        "version":1,
        "sha256Hash":"45ec52f8be059c86962446872ae55974c5a27944d9e530358a79ae19e828f549"
    }
}
```

```
{
    "errors": [
        {
            "message": "PersistedQueryNotFound",
            "extensions": {
                "code": "PERSISTED_QUERY_NOT_FOUND",
                "exception": {
                    "stacktrace": [
                        "PersistedQueryNotFoundError: PersistedQueryNotFound",
                        "    at Object.<anonymous>
(/Users/bryantebeek/Lifely/Presentations/20191103-graphql-in-production/node_modules/apollo-
server/node_modules/apollo-server-core/src/requestPipeline.ts:164:15)",
                        "    at Generator.next (<anonymous>)",
                        "    at fulfilled (/Users/bryantebeek/Lifely/Presentations/20191103-
graphql-in-production/node_modules/apollo-server/node_modules/apollo-server-
core/dist/requestPipeline.js:5:58)",
                        "    at process._tickCallback (internal/process/next_tick.js:68:7)"
                    ]
                }
            }
        }
    ]
}
```

```
// http://localhost:4000/graphql?
extensions=%7B%0A%20%20%20%20%22persistedQuery%22%3A%20%7B%0A%20%20%20%20%20%20%20%20%22versi
on%22%3A1,%0A%20%20%20%20%20%20%20%20%22sha256Hash%22%3A%2245ec52f8be059c86962446872ae55974c5
a27944d9e530358a79ae19e828f549%22%0A%20%20%20%20%7D%0A%7D&query=query%20%7B%0A%20%20%20%20peo
ple%20%7B%0A%20%20%20%20%20%20%20%20id%0A%20%20%20%20%20%20%20%20name%0A%20%20%20%20%7D%0A%7D



// extensions=
{
    "persistedQuery": {
        "version":1,
        "sha256Hash":"45ec52f8be059c86962446872ae55974c5a27944d9e530358a79ae19e828f549"
    }
}

// query=
query {
    people {
        id
        name
    }
}
```

Demo 4: Persisted queries.

Security in GraphQL.

- Limiting your queries
- Disabling the introspection query
- Masking your errors
- Enforcing pagination
- Rate-limiting

Limiting your queries.

- Limiting on query depth
- Limiting on query complexity
- Persisted queries (whitelisting) (no demo)

Demo 5.1+5.2:
Limiting your queries.

Disabling the introspection query.

- Very useful in development
- Provides autocompletion for GraphiQL, Insomnia, etc.
- Disable for production environments
- Apollo server disables it by default if

  process.env.NODE_ENV === 'production'

Demo 5.3: Disable the introspection query.

Masking errors.

- Exposes all errors to the client by default
  - Including stack traces (disabled on production)
- Whitelist which errors are allowed
  - e.g. Input validation errors

Demo 5.4: Masking errors.

Enforcing pagination.

- Make sure you paginate all your object types
- Always set a ceiling on the amount a user can retrieve

Demo 5.5: Enforcing pagination.

Rate limiting.

- Prevent DoS attacks
- https://github.com/nfriedly/express-rate-limit (express)
- https://github.com/teamplanes/graphql-rate-limit (graphql)

Demo 5.6: Rate limiting.

Graph traversal "attack".

- Becomes harder to prevent the larger the graph becomes
- Authorization not only at object level, but also at field level

Demo 6: Graph traversal.

Bryan te Beek
Lead Backend / Ops

bryan@lifely.nl

@bryantebeek

bryantebeek/graphql-in-production

lifely