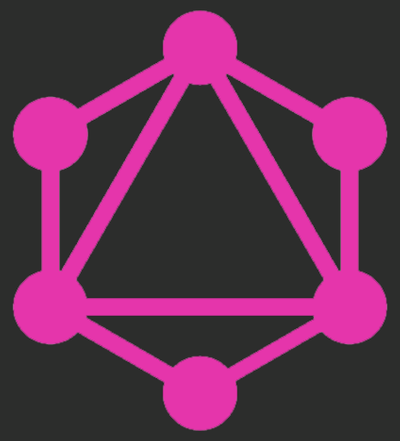


GraphQLR

A DATA QUERY LANGUAGE AND RUNTIME

Barret Schloerke
Statistics PhD Candidate
Purdue University

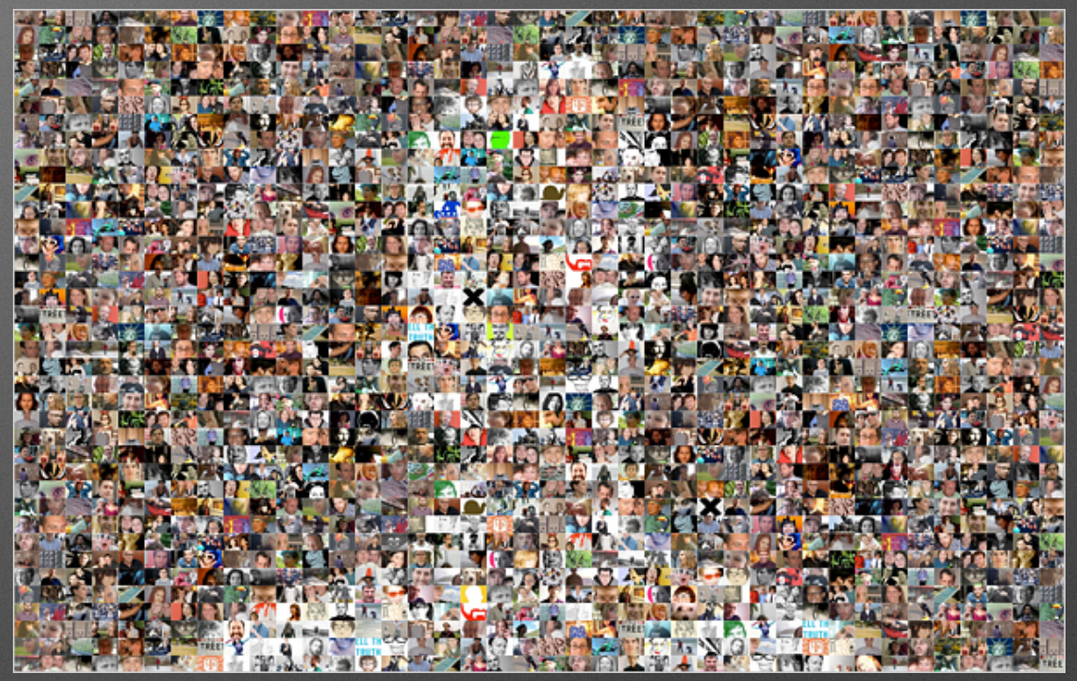
NSF Grant: DGE-1333468

About Me

- **Purdue University**
  - 3rd Year Statistics PhD Candidate
  - Dr. William Cleveland and Dr. Ryan Hafen
  - Research in large data visualization using R - www.tessera.io
- Metamarkets.com - 1.5 years
  - Front end engineer - coffee script / node.js
- **Iowa State University**
  - B.S. in Computer Engineering
  - Research in statistical data visualization with R

Querying data from a web browser

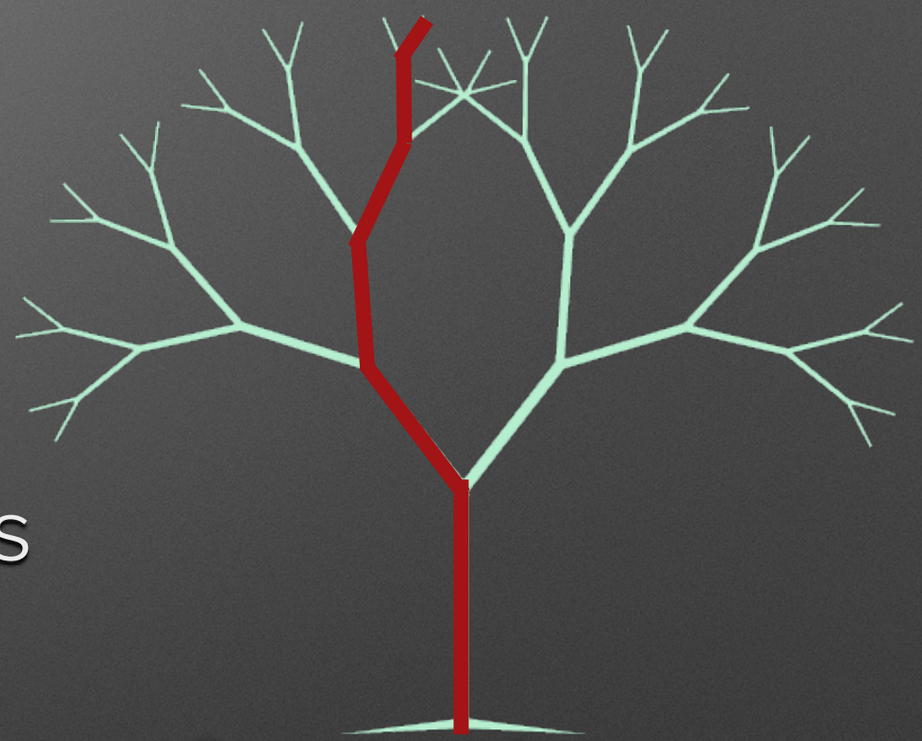
Example: Facebook Friend Info



- Display all of my friends'
  - profile picture
  - full name
- REST (naive server setup)
  - `Ask for all n friend IDs`
  - `For each friend ID:`
    - `Ask server for friend ID's profile information`
- Total query count… **`1 + n`**

Facebook Friend Info Limitations

- **n + 1** queries!
  - Browsers limited to **6-8** parallel connections per host
  - ~15 seconds to load 1001 requests at 0.1 s/request
  - only **one** part of the website!
- Bottleneck is with the data server API



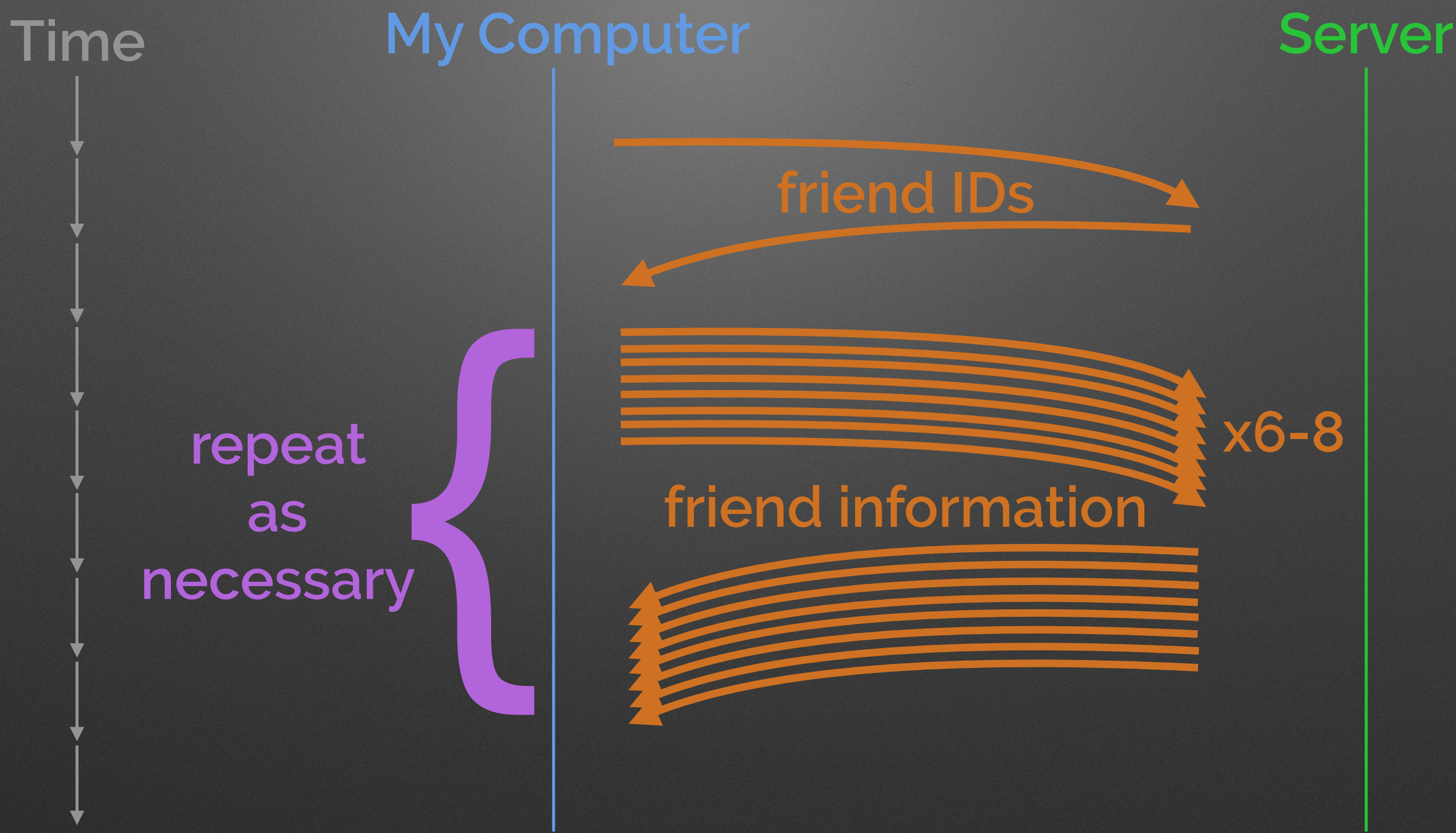
facebook.com

http://www.browserscope.org/?category=network

Data Server API Spectrum

- Naive REST (Easier)
  - Easy to implement
  - **Very slow** to execute (n + 1 queries)

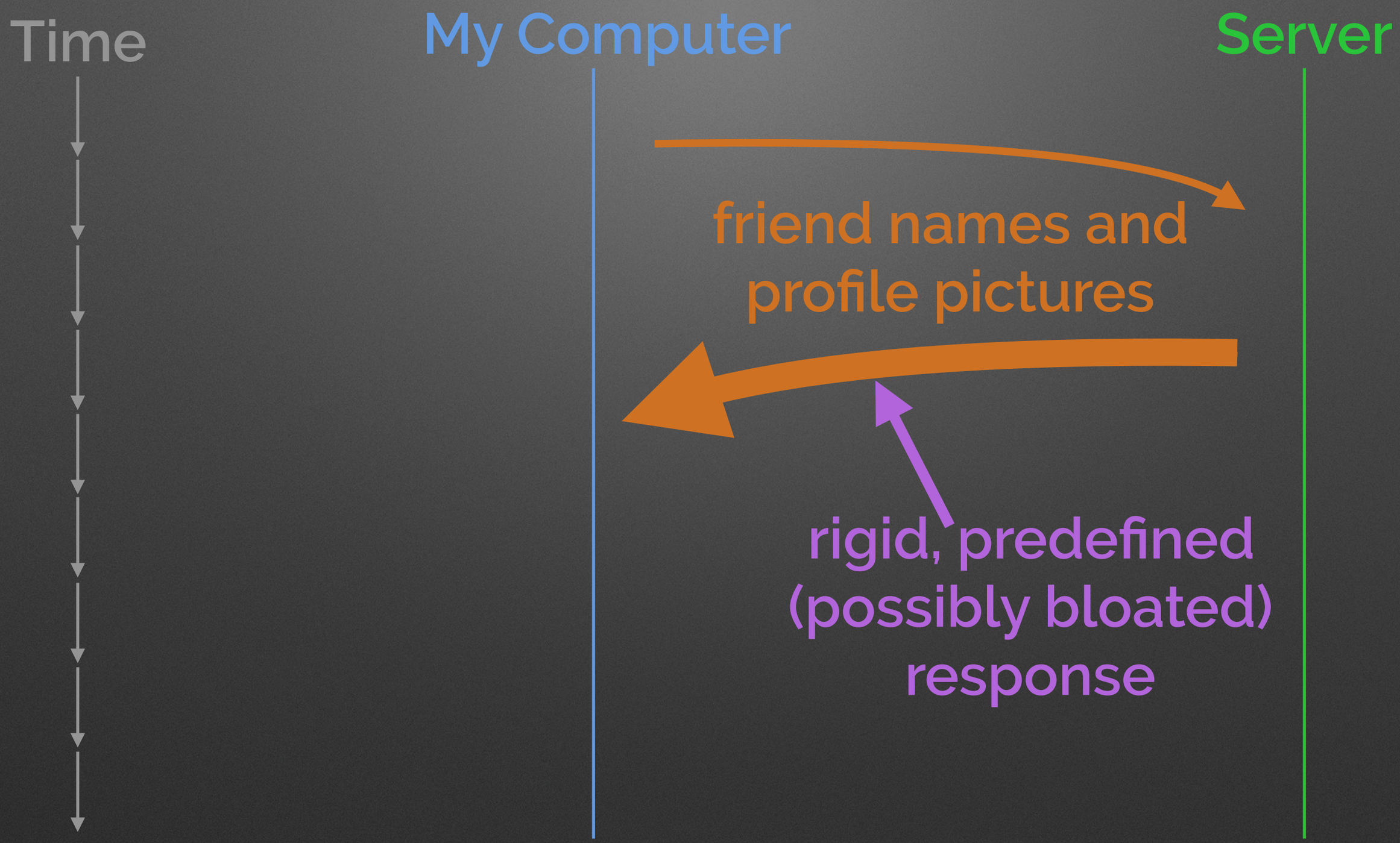
Naive REST


Time
My Computer
Server
friend IDs
repeat
as
necessary
x6-8
friend information

Data Server API Spectrum

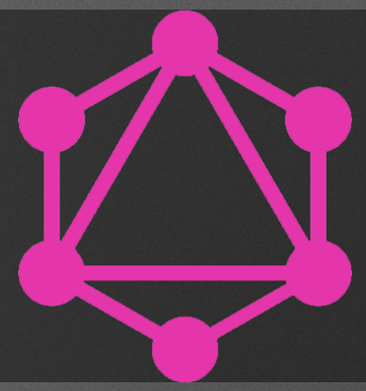
- Naive REST
  - Easy to implement
  - **Very slow** to execute (n + 1 queries)
- Custom Server
  - Difficult to implement
  - **Fast** (1 query)
  - Every browser data need is a custom server response
  - Separation of browser information needs and server information availability
    - Typically causes over-fetching of data

Custom Server



Naive + Custom Data Server API?

GraphQL



- **Graph** **Q**uery **L**anguage
  - “A data query language and runtime”
- Facebook open sourced the specification in mid 2015
- Backend agnostic data query language built upon strong-typed hierarchical sets of fields.
  - "strong type system" is described as one in which there is no possibility of an unchecked runtime type error
- “The query is shaped just like the data it returns. It is a natural way for product engineers to describe data requirements.”
  - Non-rigid
  - Avoids under-fetching and over-fetching

http://graphql.org/ https://en.wikipedia.org/wiki/Strong_and_weak_typing

Two parts

- Schema
  - Defines the strong typed objects
- Query
  - Asks for objects and fields defined in the Schema

Facebook Example: GraphQL

- Schema

```
- scalar LocalUrl

- type User {
    id: Int
    name: String
    profPic: LocalUrl
    friends: [User]
  }

- type Query {
    user(id: String!): User
  }
```

- Query

```
- query friends_info {
    user(id: 3945) {
      name,
      profPic
      friends: {
        id,
        name,
        profPic
      }
    }
  }
```

Facebook Example: Result


```
· {
    “user”: {
      “name”: “Barret”,
      “profPic”: “/p/3945”,
      “friends”: [
{“id”: 1436, “name”: “Di”,     “profPic”: “/p/1436”},
{“id”: 3849, “name”: “Rob”,    “profPic”: “/p/3849”},
{“id”: 5978, “name”: “Hadley”, “profPic”: “/p/5978”},
{“id”: 9632, “name”: “Heike”,  “profPic”: “/p/9632”},
{“id”: 2931, “name”: “Carson”, “profPic”: “/p/2931”},
…
      ]
    }
}
```

Endless Query Options

- Only restricted by Schema definition
  - User's `name` only
  - User's `name` and `profPic`
  - User's friends of friends' `id` and `profPic`

GraphQLR

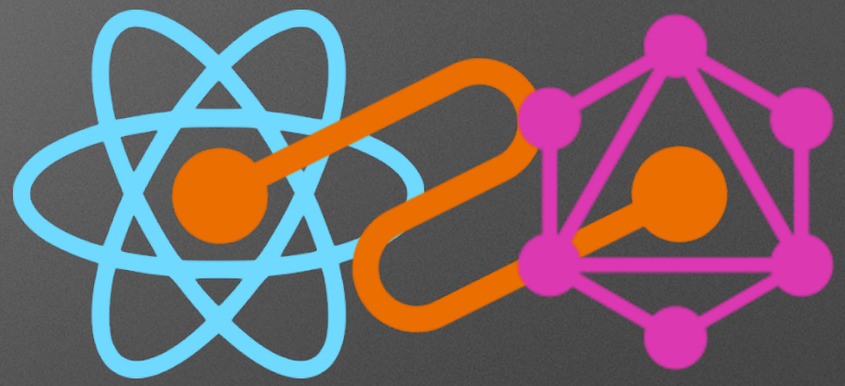
- GraphQL with the power of R
  - github.com/schloerke/graphqlr
  - Release goal: May 2016
- Retrieve data from...
  - memory / disk
  - external databases (hadoop, mysql, ...)
  - **simulation / calculation**
    - Use any R package or personal scripts!

Power of R

- ```
  type User {
    id:        Int
    name:      String
    profPic:   LocalUrl
    friends:   [User]
    bffCluster: [User]
  }
  ```
- ‘`bffCluster`’ should be calculated on the fly
  - Expensive calculation to do for **everyone** at **all times**
  - `fastcluster::hclust`
    - External script!
  ```

Immediate Uses

- relay web applications
  - https://facebook.github.io/relay/



- ex: trelliscope
  - complex R application
  - migrating from shiny to pure javascript with GraphQLR data server
  - http://tessera.io/docs-trelliscope/

Websites

- Main GraphQL Website
  - graphql.org
- Specification Document
  - facebook.github.io/graphql
- Javascript Implementation of GraphQL
  - github.com/graphql/graphql-js
- Learn GraphQL
  - github.com/dwyl/learn-graphQL

```
type Question {
  id: Int,
  question: String,
  answer: String,
  confidence: Number
}
type Query {
  question(id: Int!): Question
}
```