Analyze your data

and save the planet

@idanyliuk

Golang Paris, 24 March '19

1. The Problem

The Problem

- Mobile app for analyzing and quiering some data
- Filter, search, different statistics
- Ability to upload new data for own analysis
- Unknown number of users (thousands, for sure)

The Problem

- Code for analysing data and parsing uploaded document has been written in Go and well tested over two years
- (That's where idea to make a mobile app started, actually)

2. The Solution

The Solution

- Client/server architecture
- Relational database
- REST HTTP/JSON APi
- N containers/VMs for horizontal scaling

Classic!

- It's a good approach if you have no idea what data are you working with
- But, if you inspect your data, often you can choose different trade-offs

3. Analyzing data

The size

- In this case:
  - Database was quite small – around half a GB
  - Growing very slowly (50MB per year)

The access pattern

- In this case:
  - Reads where often (hundreds or thousands per second in a peak times)
  - Writes were extremely rare (once per week max, once per two months min)
- Assymetry!

Data cost profile

- In this case:
  - It was okay to get delays in updating latest data

4. New solution!

Squeeze data

- Normalizing - remove every repetetive string
- Switching to more memory saving format - flatbuffers
- Bitpacking - i.e. if integer is in 0..16, you can pack 2 ints per byte

Reduced 500MB dataset to 13(!) MB flatbuffers file

Good enough to ship it with mobile app itself!

New Solution

- Ship dataset with mobile app itself
- Use Gomobile to reuse existing code for parsing and analysing data
- Run all searches, queries & stats on the phone
- Update app in stores once per month, when the dataset is updated (week delay)

New Solution

- Use Flutter + Gomobile ultra combo
- Implement statistics and data analysis on the device
- NO SERVERS

New Solution

- Use Flutter + Gomobile ultra combo
- Implement statistics and data analysis on the device
- NO SERVERS **(LITERALLY, ZERO)**

Results

Results

- Massive decrease of amount of servers :)
- No need to deploy and scale servers
- No need to setup observability stack (nothing to observe)
- Saved money -$$$

Results

- Three orders of magnitude faster user experience (memory fetch vs network call)
- App works offline! :)
- Still Go, so most of the logic can be tested and benchmarked with awesome Go tooling

Results

- Zero energy consumption at data center :)

The best code is no code

github.com/kelseyhightower/nocode

The best distributed system is no distributed system

Thank you