Speed up the monolith
building a smart reverse proxy in Go
Alessio Caiazza
Senior Backend Engineer, Infrastructure

@nolith - alessio@gitlab.com
We are a Ruby shop

Why Go?
Slow requests
unicorn

Rack HTTP server for fast clients and Unix

unicorn is an HTTP server for Rack applications designed to only serve fast clients on low-latency, high-bandwidth connections and take advantage of features in Unix/Unix-like kernels.

— bogomips.org/unicorn
Workhorse
a smart reverse proxy
package main

import (  "log"  "net/http"  "net/http/httputil"  "net/url"
)

func main() {  upstream, err := url.Parse("https://httpbin.org")  if err != nil {    log.Fatal(err)  }

  proxy := httputil.NewSingleHostReverseProxy(upstream)
  err = http.ListenAndServe(":8080", proxy)
  log.Fatal(err)  }
Speed up a slow request
func main() {
    r := mux.NewRouter() // import "github.com/gorilla/mux"
    r.Use(proxyHeadersMiddleware)
    r.HandleFunc("/slow", func(w http.ResponseWriter, r *http.Request) {
        w.Write([]byte("Now faster!"))
    })

    upstream, err := url.Parse("https://httpbin.org")
    if err != nil {
        log.Fatal(err)
    }

    proxy := httputil.NewSingleHostReverseProxy(upstream)
    r.PathPrefix("/").Handler(proxy)

    err = http.ListenAndServe(":8080", r)
    log.Fatal(err)
}

func proxyHeadersMiddleware(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        r.Header.Set("X-Forwarded-Proto", r.URL.Scheme)

        next.ServeHTTP(w, r)
    })
}
Speed up git

电商平台到

Slow request
Body Hijacking
POST /some/url/upload

save the incoming file on a temporary location

file was replaced with its location and other metadata

POST /some/url/upload

request result

cleanup
Speed up uploads

👉 Slow request
Cloud native charts and Network File System
HA Proxy

GitLab

Redis

Sidekiq

enqueue a job

dequeue a job
HA Proxy

GitLab

Sidekiq

Redis

NFS

enqueue a job

dequeue a job
Object Storage in Workhorse

👋 NFS
git LFS
io package
io.Reader | io.Writer = ❤
// install route with r.Handle("/upload/{file}", body.HijackHandler(proxy)).Methods("PUT")
func HijackHandler(next http.Handler) http.Handler {
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
        presigned, err := api.AuthorizeUpload(r)
        if err != nil {
            http.Error(w, err.Error(), 503)
            return
        }
        if err != nil {
            http.Error(w, err.Error(), 500)
            return
        }
        upload.ContentLength = r.ContentLength
        upload.Header.Set("X-My-File-Path", presigned.Path) // sign this
        resp, err := http.DefaultClient.Do(upload)
        if err != nil {
            http.Error(w, err.Error(), 503)
            return
        }
        defer resp.Body.Close()
        if resp.StatusCode != http.StatusOK {
            http.Error(w, resp.Status, 503)
            return
        }
        toProxy := r.Clone(r.Context())
        toProxy.Body = nil
        toProxy.ContentLength = 0
        toProxy.Header.Set("X-My-File-Path", presigned.Path) // sign this
        next.ServeHTTP(w, toProxy)
    })
}
Mission Complete!
Mission Complete!
Not exactly
Unknown length requests from ~35k CI runners
Multipart Upload
divide and upload

Original Object
Delineated Into Parts
Parts Uploaded to S3
Object Finalized in S3
Keep memory usage under control

process a chunk at time

Create a temp file
Write up to 50 MB
Upload temp file
Delete temp file
EOF?
true
Finalize upload
Thanks!

✴ You can speed up a web application writing a reverse proxy in Go
✴ An iterative approach
✴ Rewrite only slow endpoints
✴ Forward to another service if needed
✴ Sign modified requests

Workhorse source code is available at gitlab.com/gitlab-org/gitlab-workhorse under MIT license.

@nolith - alessio@gitlab.com