Как обрабатывать 1TB гистологических иссл едований в день

Евгений Конечный
Uzum Tezkor

Кто я?

Роль:

- Team Lead в **Uzum Tezkor**
- Co-Founder в **Histoscan**

Опыт:

- Golang: 7 лет
- TL: 5 лет
- FoodTech & MedTech

TL;DR

О чем буду говорить:

- Что такое Digital Pathology и какие там есть челленджи
- Что такое Whole Slide Image?
- Openslide + Go
- Реализация Deep Zoom Generator
- Оптимизация процессинга изображений

TL;DR

О чем буду говорить:

- Что такое Digital Pathology и какие там есть челленджи
- Что такое Whole Slide Image?
- Openslide + Go
- Реализация Deep Zoom Generator
- Оптимизация процессинга изображений

О чем НЕ буду говорить:

- Оптимизация хранения и алгоритмы сжатия
- Инфраструктурные решения с SSD Caching, типа ZFS, bcachefs и тп
- Ни слова про AI

Экспресс-курс в Digital Pathology

Pathology

Чем занимаются патологоанатомы

- Определяют причины смерти
- Производят вскрытия
- Другой мрачняк 💀

Pathology

Чем занимаются патологоанатомы

- Определяют причины смерти
- Производят вскрытия
- Другой мрачняк 💀

Чем НАС САМОМ ДЕЛЕ занимаются патологоанатомы

- Исследование гистологических срезов тканей живых людей, чтобы поставить им диагноз
- 90% работы – онкология

Histology

Процесс гистологического исследования

- Забор биопсийного материала
- Дегидратация материала
- Залить в парафиновый блок
- Нарезка на слайсы толщиной несколько микрон
- Окраска
- Помещение под стекло
- Изучение ткани в микроскоп

GoFunc

Histology

Основная цель
Написать экспертное заключение

Основной способ диагностики раньше
Диагностика через микроскоп

Основной способ диагностики сейчас
Сканируем и диагностируем за монитором

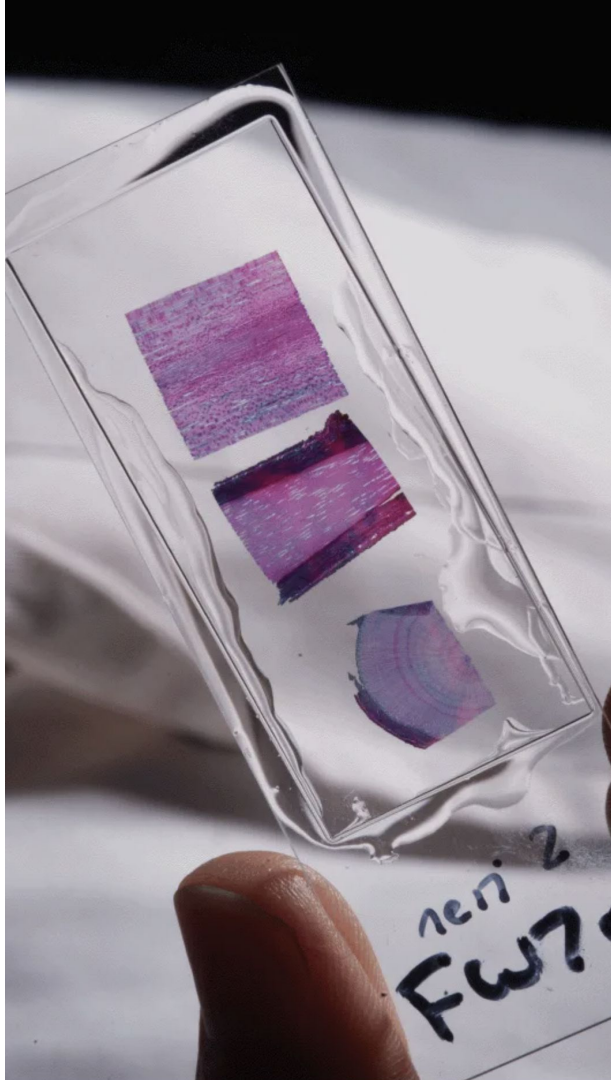


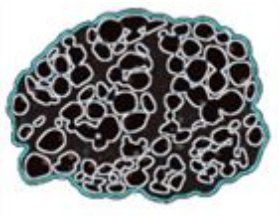
Image Analysis

Tissue-level

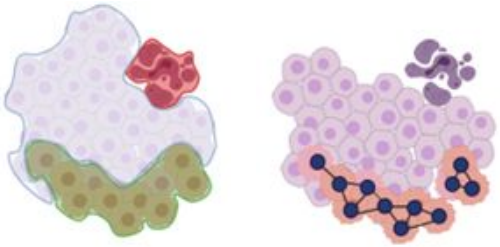
GoFunc

Image Analysis

Tissue-level



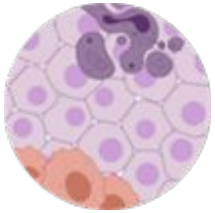
Object-level



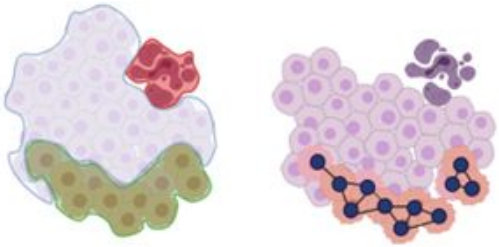
GoFunc

Image Analysis

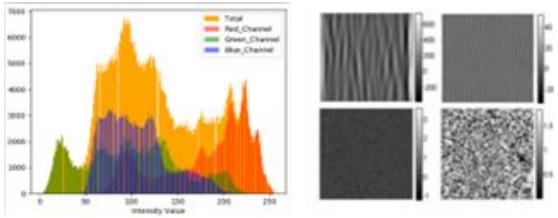
Tissue-level



Object-level



Pixel-level



GoFunc

Computational pathology

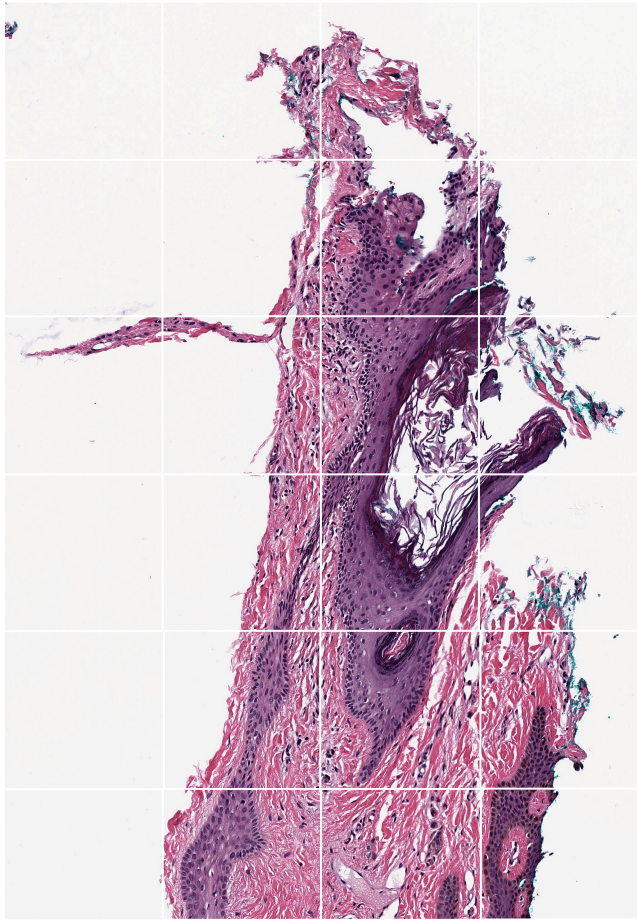
https://www.sciencedirect.com/science/article/pii/S2153353923001712

GoFunc

Challenges

Петабайты данных

- 1000 сканирований в день
- Размер файла ~ 1.7GB
- TTL файла от 1 до 5 лет



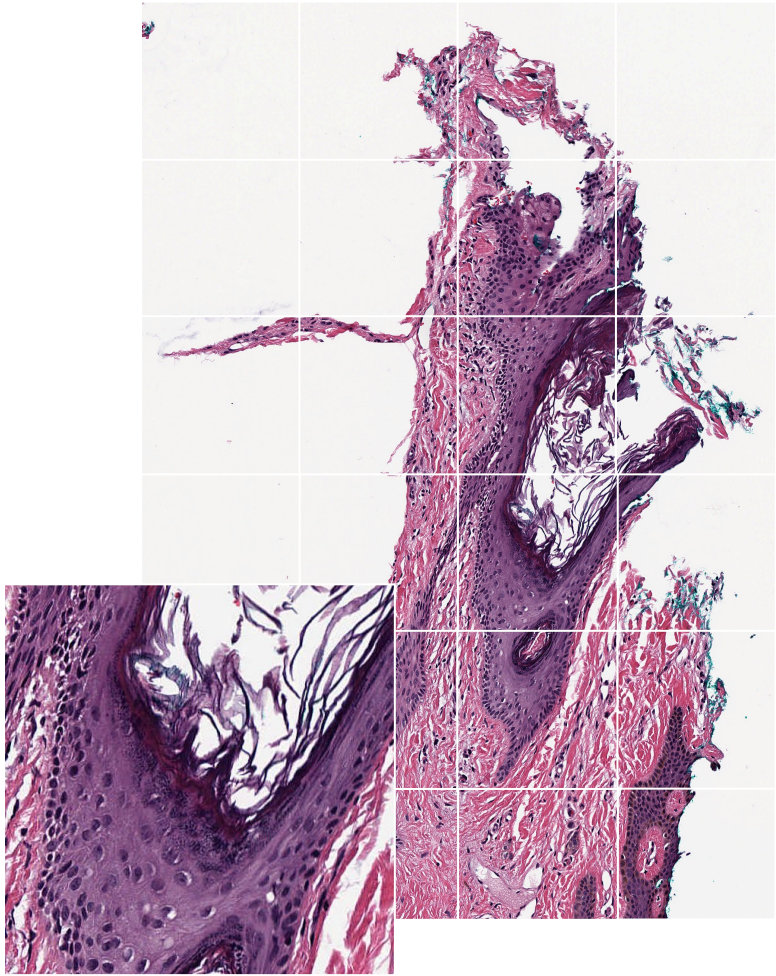
Challenges

Петабайты данных

- 1000 сканирований в день
- Размер файла ~ 1.7GB
- TTL файла от 1 до 5 лет

Скорость доступа

- Врач не должен видеть артефактов при просмотре изображения
- 95% Latency ≤ 50ms
- Зависимость от I/O



Challenges

Нельзя выбрать три из трех

- Долговременное надежное хранилище на ~ **5PT**
- Высокая скорость доступа
- Дешево

Challenges

Нельзя выбрать три из трех

- Долговременное надежное хранилище на ~ **5PT**
- Высокая скорость доступа
- Дешево

Или можно это делать не одновременно?

- Подготавливать оригиналы изображений
- Использовать горячие и холодные хранилища

GoFunc

Hot & Cold Storage

Hot & Cold Storage

Что это?

- Cold Storage – Долговременное файловое хранилище с низким I/O

GoFunc

Hot & Cold Storage

Что это?

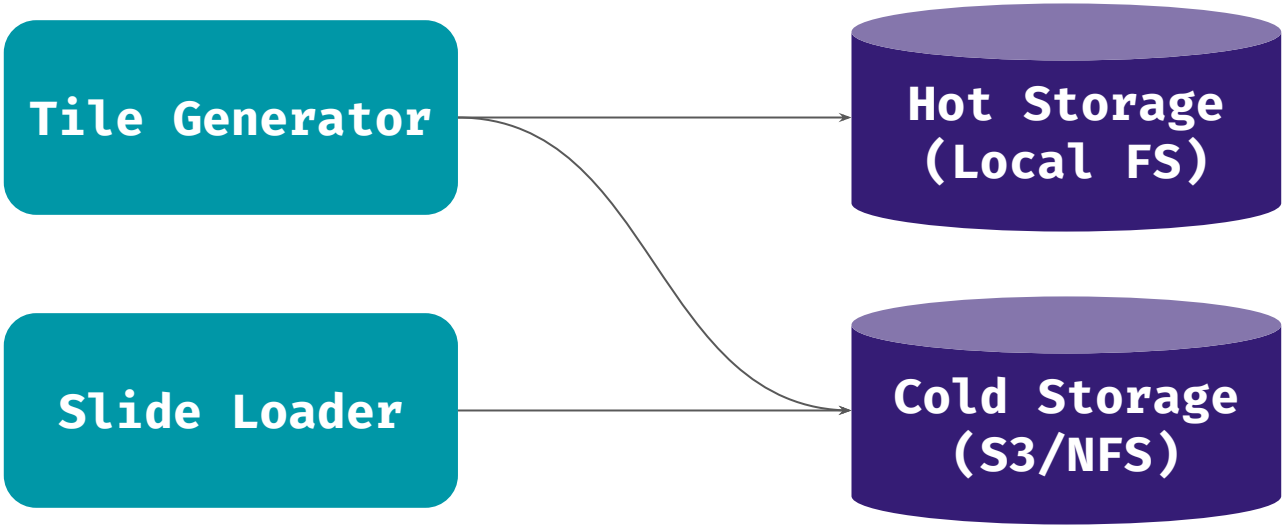
- **Cold Storage** – Долговременное файловое хранилище с низким I/O
- **Hot Storage** – Небольшое, но очень производительное хранилище ⚡

Hot & Cold Storage

Что это?

- **Cold Storage** – Долговременное файловое хранилище с низким I/O
- **Hot Storage** – Небольшое, но очень производительное хранилище ⚡
- Все файлы лежат на **Cold**, но во время работы переносятся на **Hot**, а после очищаются

Hot & Cold Storage



GoFunc

А как хранить то?

Save to...

Во что конвертить?

➔ Контейнерные форматы – DICOM SM, TIFF, OME-NGFF

GoFunc

Save to...

Во что конвертить?

- Контейнерные форматы – DICOM SM, TIFF, OME-NGFF
- ➔ Базы для хранения тайлов – MBTiles, TileDB, etc

GoFunc

Save to...

Во что конвертить?

- Контейнерные форматы – DICOM SM, TIFF, OME-NGFF
- Базы для хранения тайлов – MBTiles, TileDB, etc

➔ ... просто тайлы, которые запросит врач?

GoFunc

Save to...

Важно

➔ По данным OSM просматривают только **1.79% всех тайлов** на карте

Самый популярный тайл в мире →

GoFunc

Save to...

Важно

- По данным OSM просматривают только **1.79% всех тайлов** на карте

➔ Врачи не лучше, они никогда не посмотрят все тайлы в одном изображении

Самый популярный тайл в гистологии →
(он прозрачный)

GoFunc

Challenges

Важно

- По данным OSM просматривают только **1.79% всех тайлов** на карте
- Врачи не лучше, они никогда не посмотрят все тайлы в одном изображении

Оптимально

- Хранить в едином формате – **DICOM SM**
- Иметь временный кэш тайлов для активных изображений

GoFunc

Whole Slide Image

Whole Slide Image

WSI:

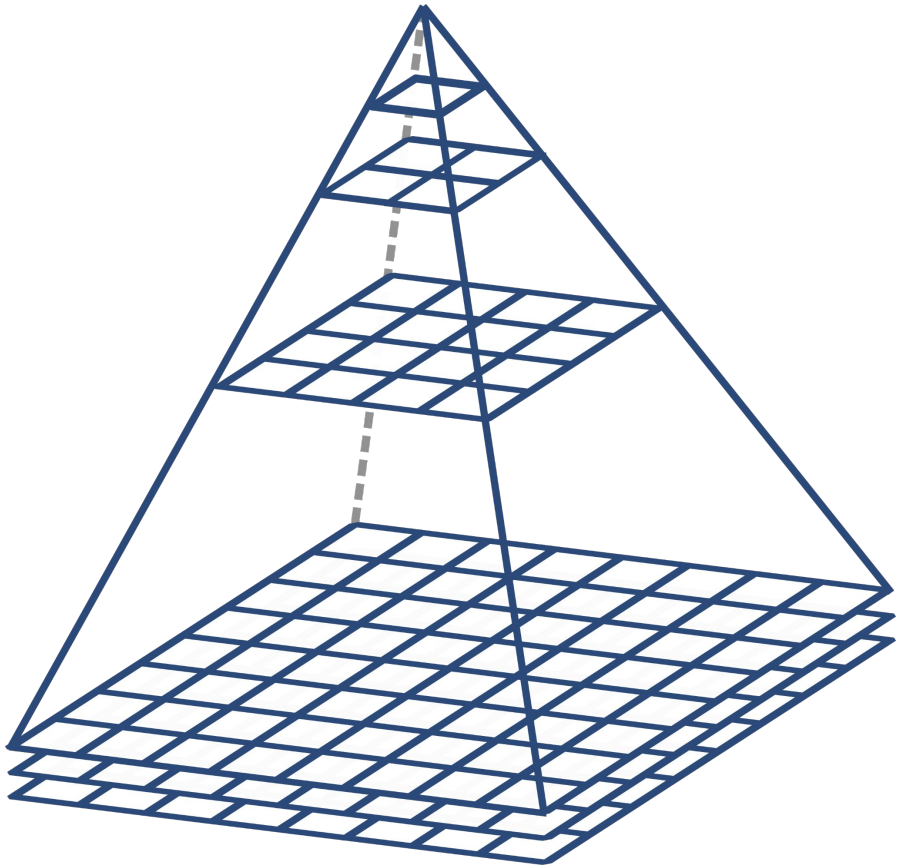
- Содержат все, что можно увидеть в микроскоп
- Представляют собой файл в виде контейнера, где:
  - Лежат несколько изображений с разным magnification
  - Может быть любой формат сжатия
  - Может быть любая специфика вендора

GoFunc

Whole Slide Image

Thumbnail

- Single Frame
- Single Object



GoFunc

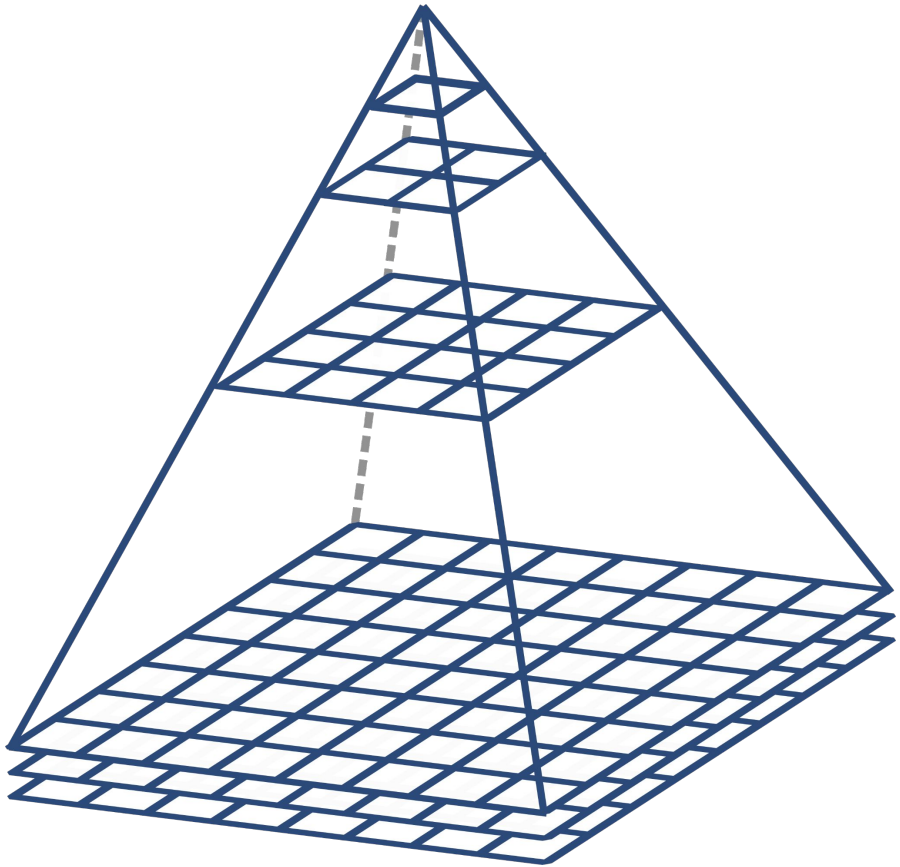
Whole Slide Image

Thumbnail

- Single Frame
- Single Object

Intermediate

- Multi frame
- Single object



GoFunc

Whole Slide Image

Thumbnail

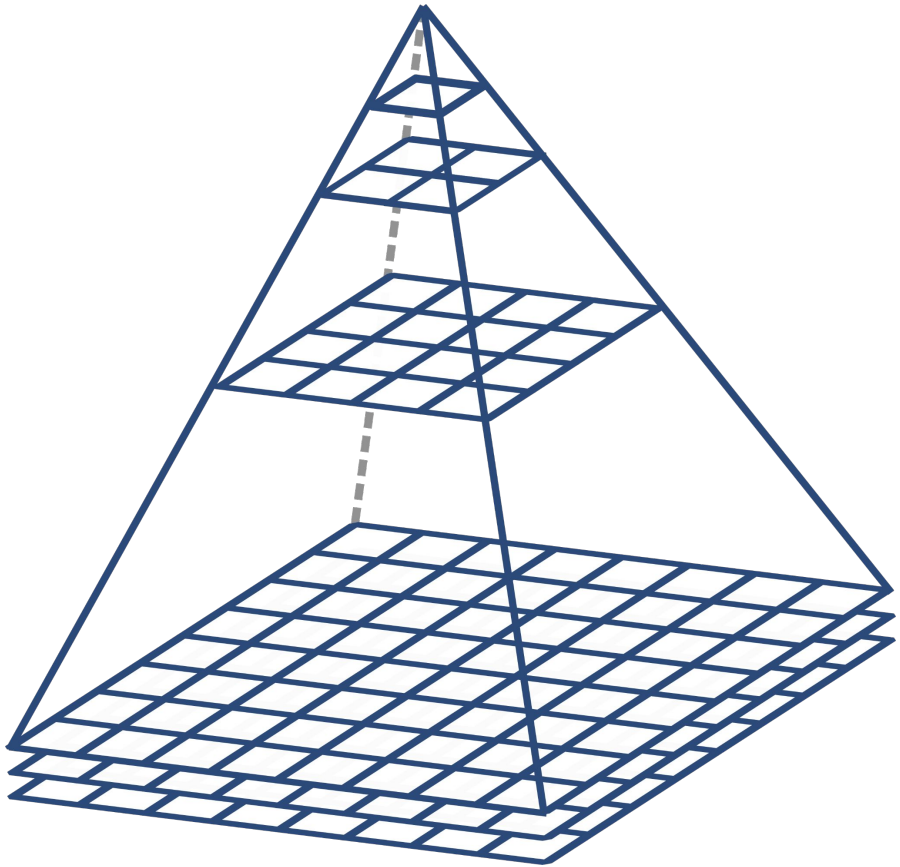
- Single Frame
- Single Object

Intermediate

- Multi frame
- Single object

Baseline

- Multi frame
- Multi object
  (z-plane/color plane)

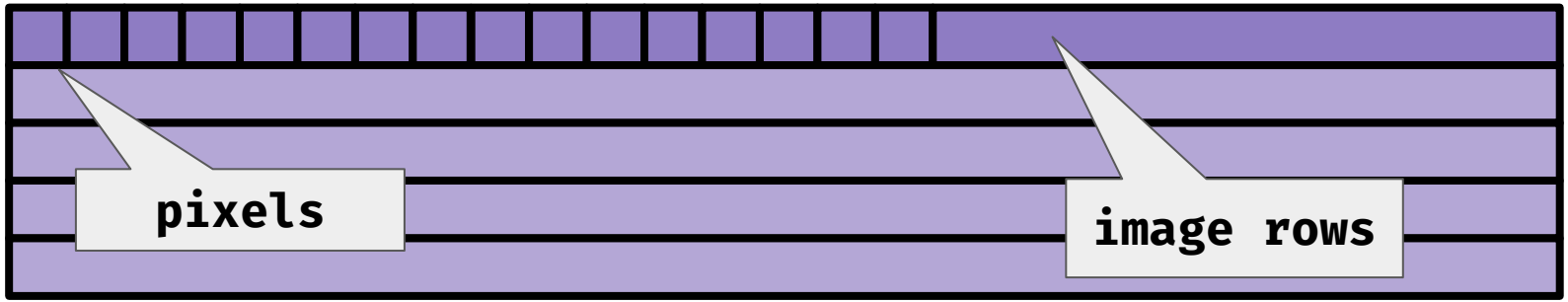


GoFunc

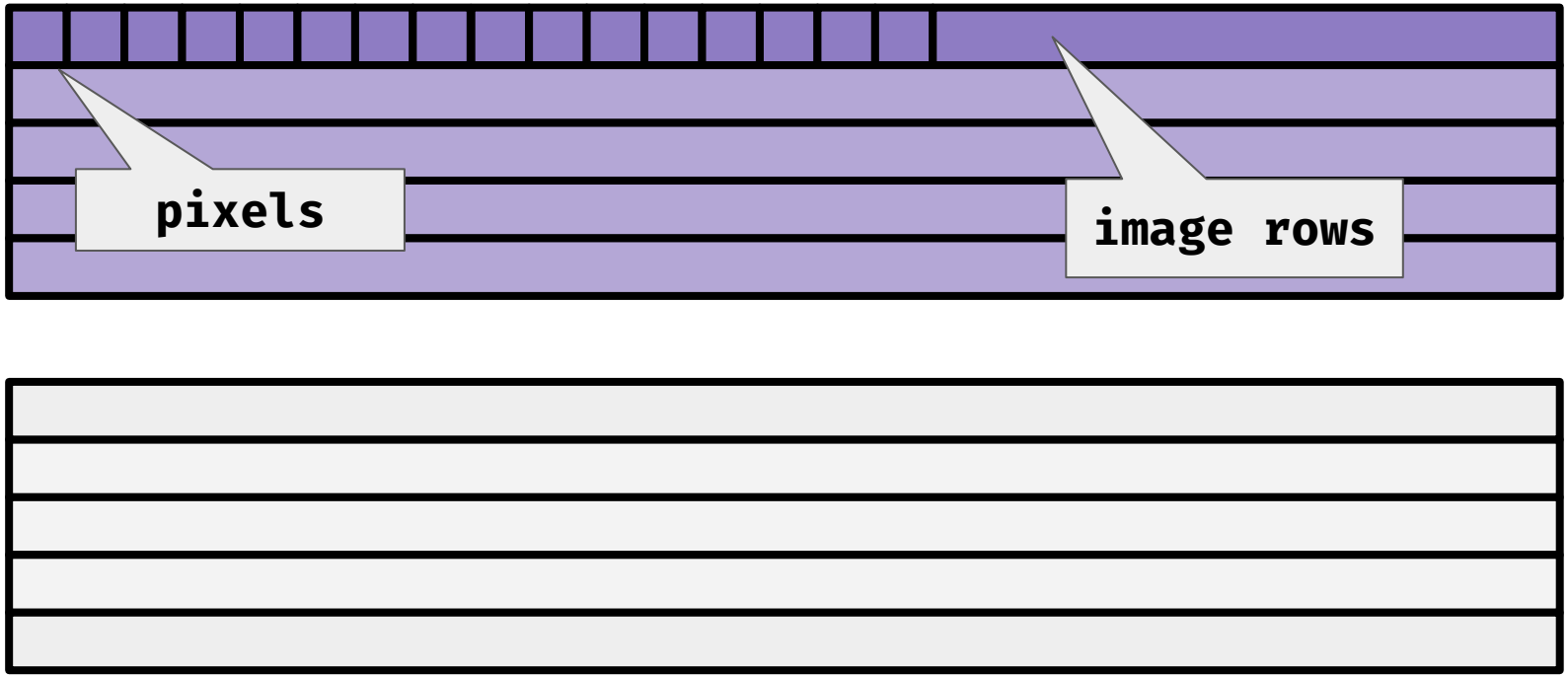
Single Frame Image

GoFunc

Single Frame Image

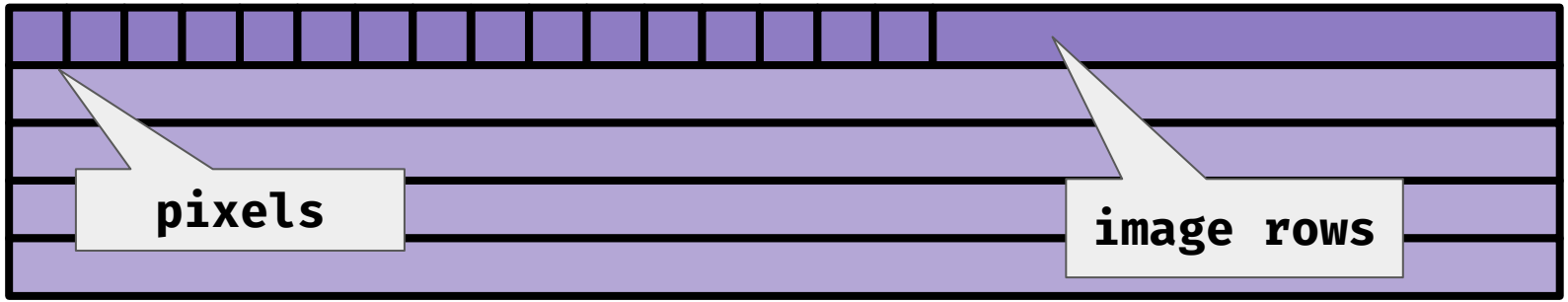


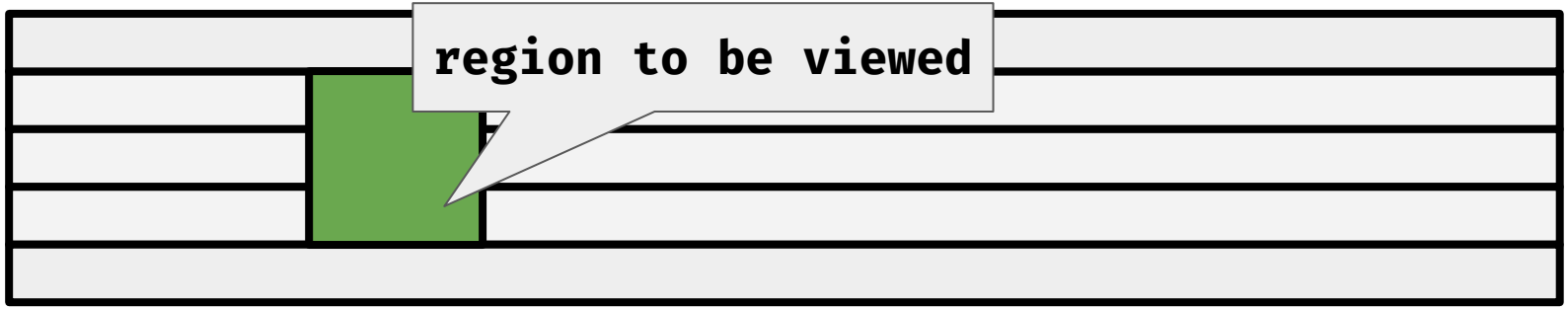
Single Frame Image



GoFunc

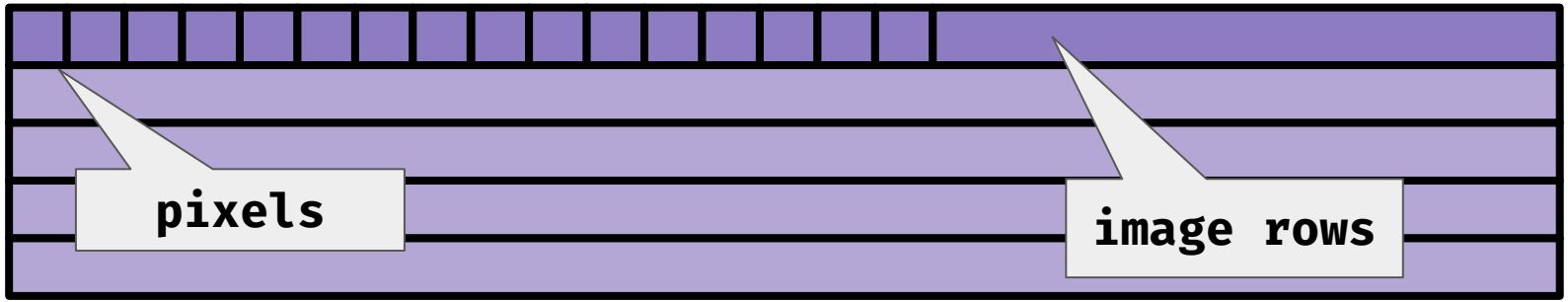
Single Frame Image

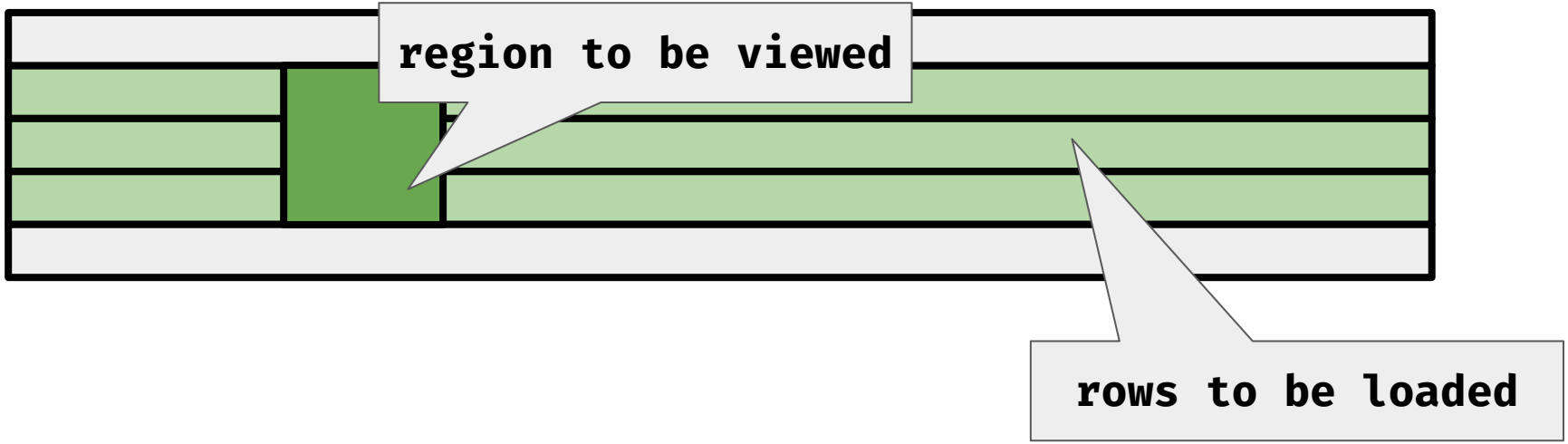




GoFunc

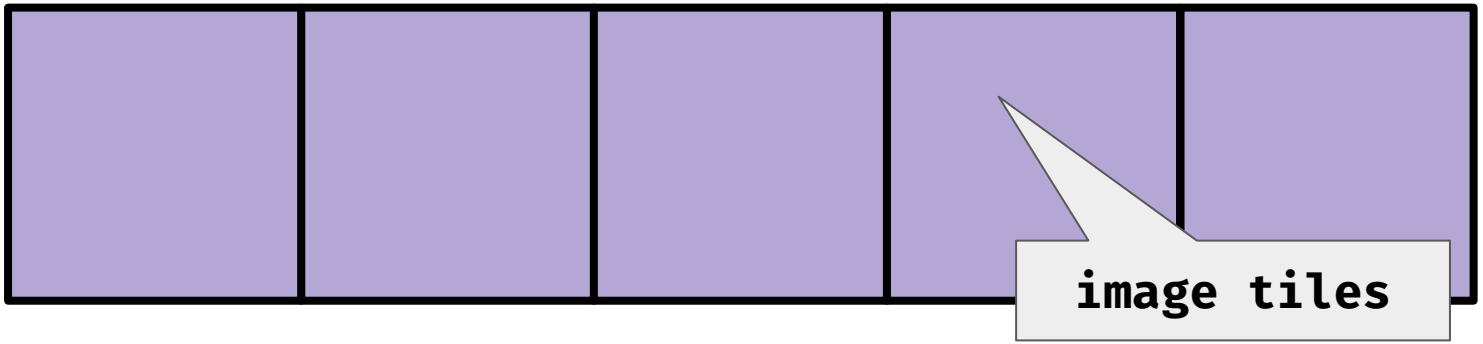
Single Frame Image





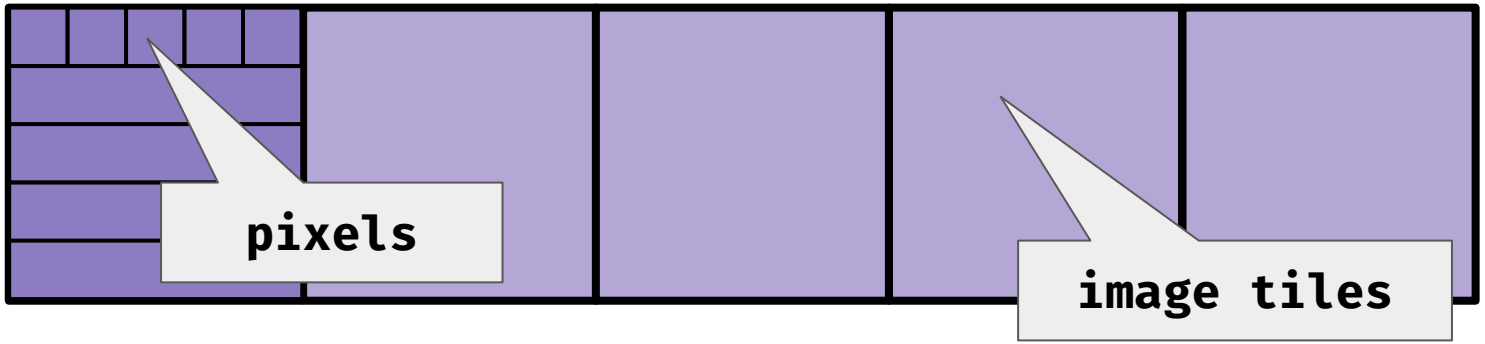
GoFunc

Multi Frame Image



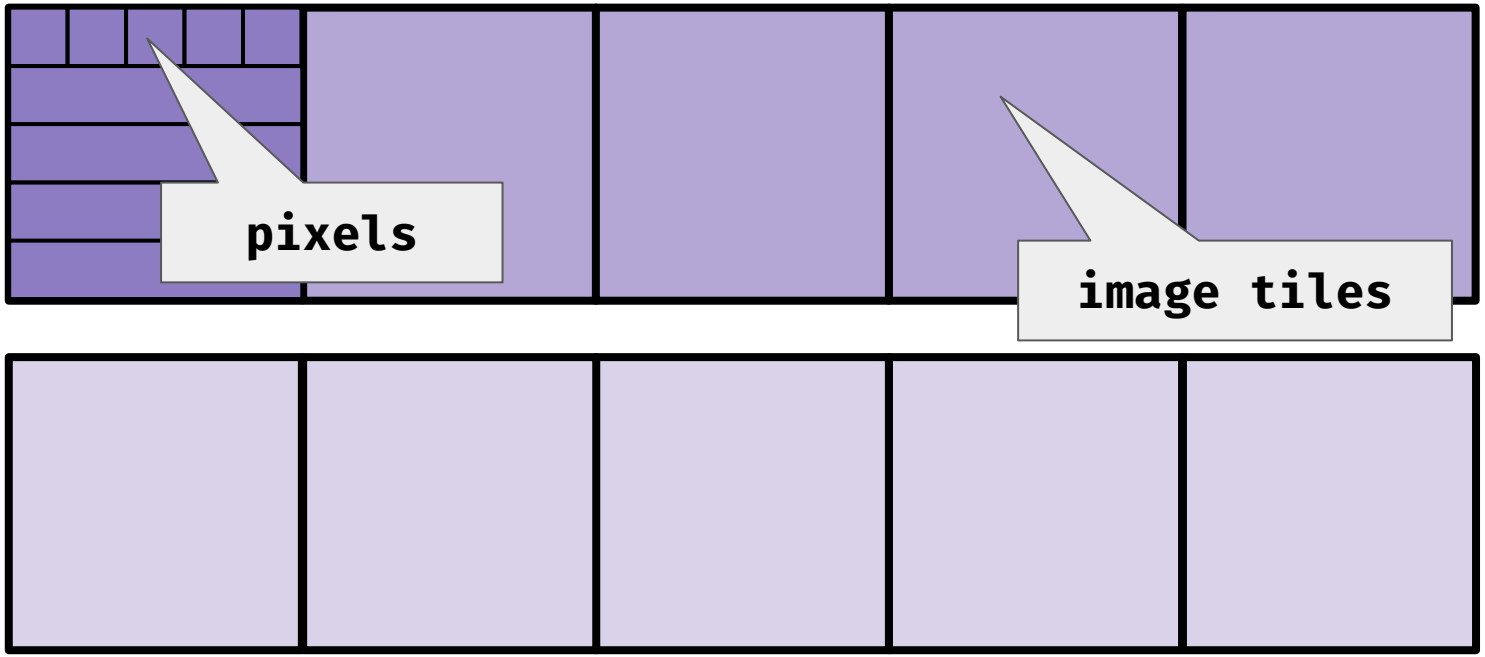
GoFunc

Multi Frame Image



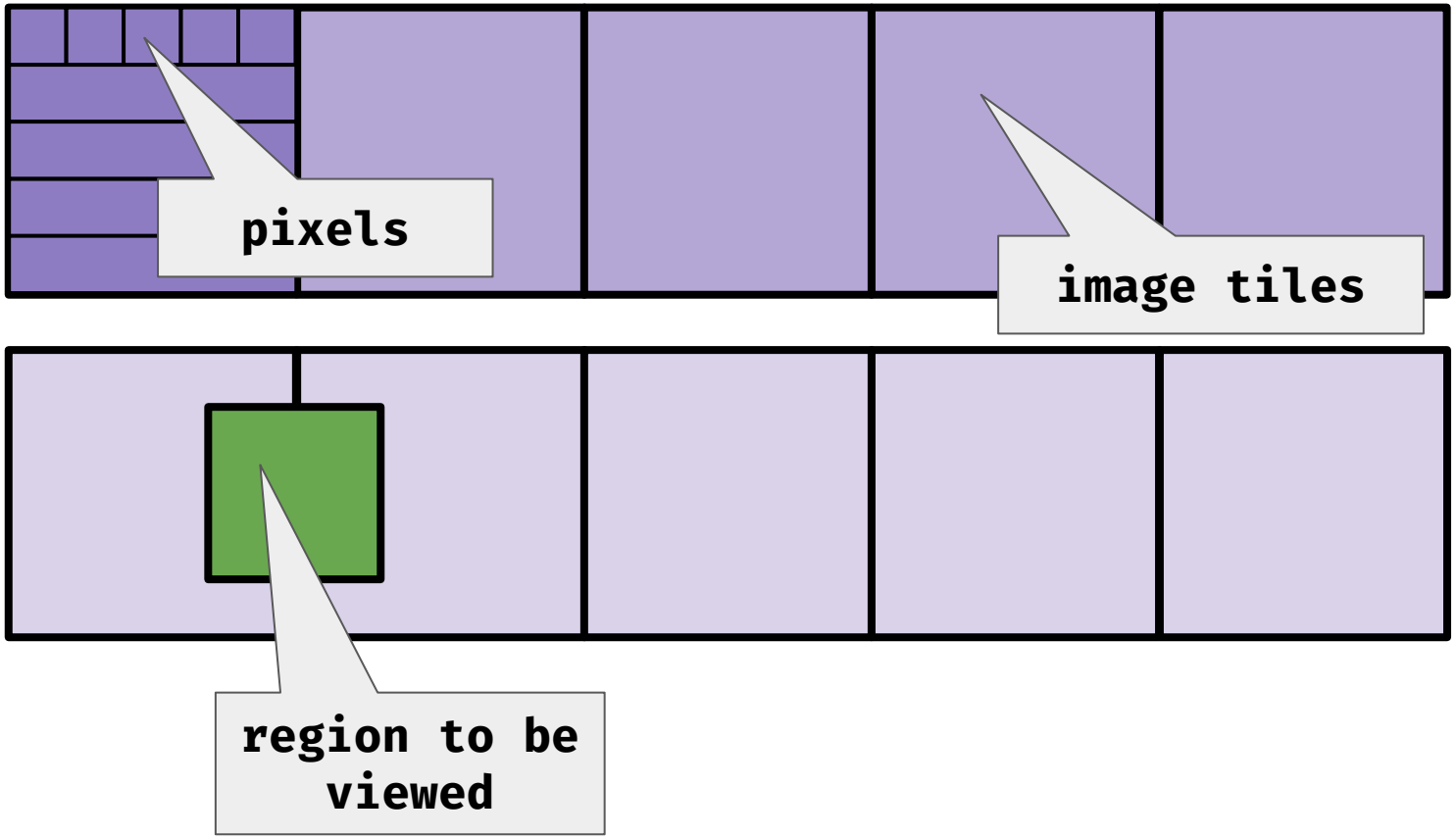
GoFunc

Multi Frame Image



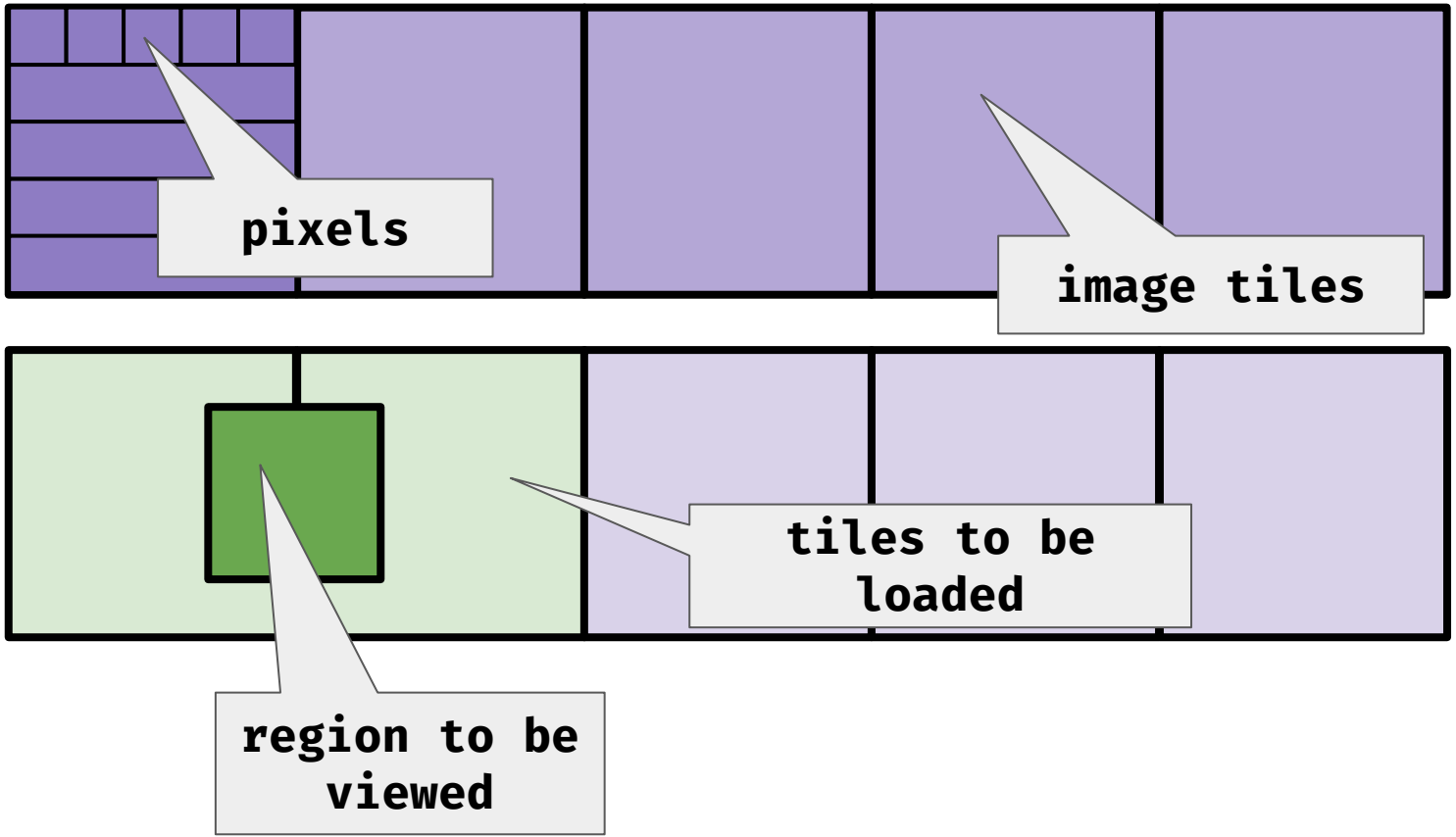
GoFunc

Multi Frame Image



GoFunc

Multi Frame Image



GoFunc

Whole Slide Image

Proprietary File Formats

- Aperio: *.svs, *.afi
- Hamamatsu: *.ndpi, *.ndpis
- Leica: *.scn
- Mirax: *.mrxs
- Ventana: *.bif, *.tif

GoFunc

Whole Slide Image

Proprietary File Formats

- Aperio: *.svs, *.afi
- Hamamatsu: *.ndpi, *.ndpis
- Leica: *.scn
- Mirax: *.mrxs
- Ventana: *.bif, *.tif

Universal format

- TIFF 😢
- DICOM SM

GoFunc

Whole Slide Image

Proprietary File Formats

- Aperio: *.svs, *.afi
- Hamamatsu: *.ndpi, *.ndpis
- Leica: *.scn
- Mirax: *.mrxs
- Ventana: *.bif, *.tif

Universal format

- TIFF 😢
- DICOM SM

Есть только **Openslide** и **Open Microscopy**

GoFunc

Openslide vs OME

Openslide

- Написан на C
- Старше всех
- Фокус на PFF

GoFunc

Openslide vs OME

Openslide

- Написан на C
- Старше всех
- Фокус на PFF

Open Microscopy Environment

- Написан на JAVA
- Современный формат, который развивается, но...
- Фокус на универсальном формате

Openslide

Почему мы форкнули Openslide

- Исправили ряд багов, которые у нас стреляли
- Расширили поддержку DICOM
- Расширили поддержку Color Plane/Z-Plane

Openslide + Go

Go Bindings


```go
package openslidego

// #cgo CFLAGS: -I/usr/include/openslide
// #cgo LDFLAGS: -lopenslide
// #include <stdio.h>
// #include <stdlib.h>
// #include <stdint.h>
// #include <openslide.h>
import "C"
```

GoFunc

Go Bindings


```go
type WSI struct {
  ptr *C.openslide_t
}
```

Go Bindings


```go
type WSI struct {
  ptr *C.openslide_t
}

func Open(filename string) (WSI, error) {
  cFilename := C.CString(filename)
  defer C.free(unsafe.Pointer(cFilename))
  data := C.openslide_open(cFilename)
  if data == nil {
     return WSI{nil}, errUnrecognized
  }
  return WSI{data}, nil
}
```

GoFunc

Go Bindings


```go
type Slide interface {
  LevelCount() int32
  LevelDimensions(level int32) (int64, int64)
  ReadRegion( ... ) (image.Image, error)
  BestLevelForDownsample(downsample float64) int32
}
```

GoFunc

Go Bindings


```go
type Slide interface {
  LevelCount() int32
  LevelDimensions(level int32) (int64, int64)
  ReadRegion( ... ) (image.Image, error)
  BestLevelForDownsample(downsample float64) int32
}

// пример реализации метода
func (s slide) LevelCount() int32 {
  return int32(C.openslide_get_level_count(s.ptr))
}
```

GoFunc

Go Bindings

Итого

- Получили биндинги для работы с Openslide
- Но не получили кода для преобразования

👀 **Посмотреть код**:

https://github.com/ekonechny/gopenslide

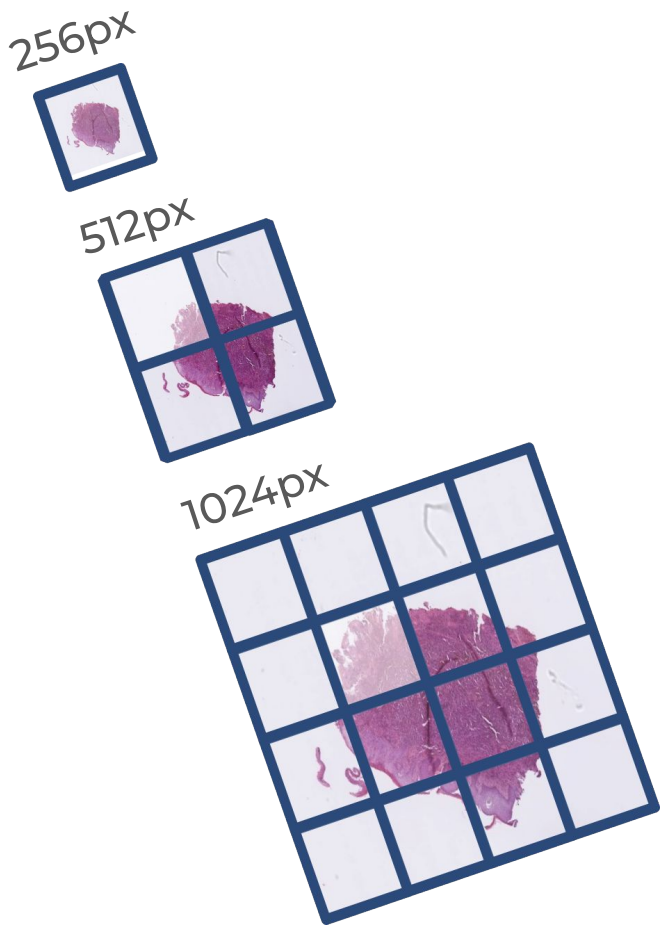
GoFunc

Deep Zoom Generator

Deep Zoom

DZI

- Спецификация для просмотра HQ-изображений
- Необходим для загрузки изображения в виде тайлов
- Zoom с DZI не соответствует Zoom с WSI
- Каждый следующий уровень в пирамиде тайлов в два раза больше следующего

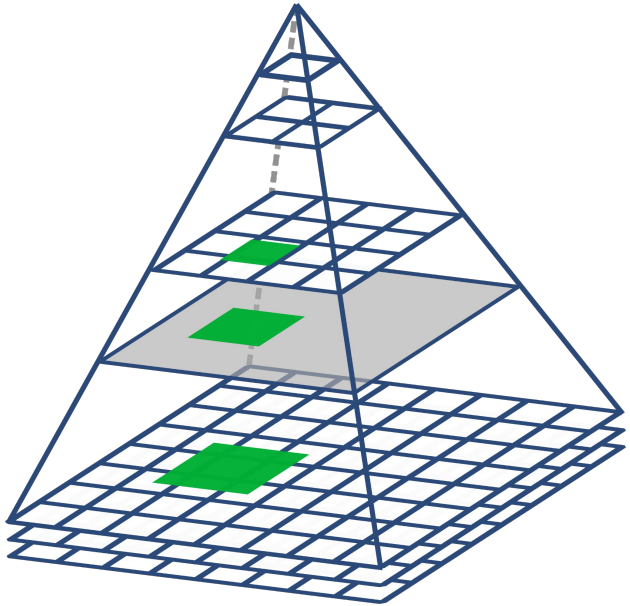


GoFunc

Deep Zoom Generator

Как получить все тайлы

➔ Получаем все уровни разрешения в исходном файле

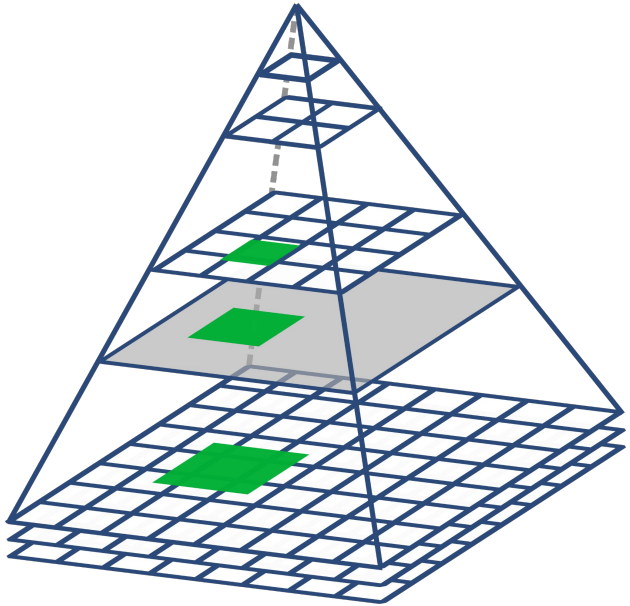


GoFunc

Deep Zoom Generator

Как получить все тайлы

- Получаем все уровни разрешения в исходном файле
- ➔ Вычисляем “виртуальные” уровни – Deep Zoom Levels и определяем количество тайлов на каждом из них



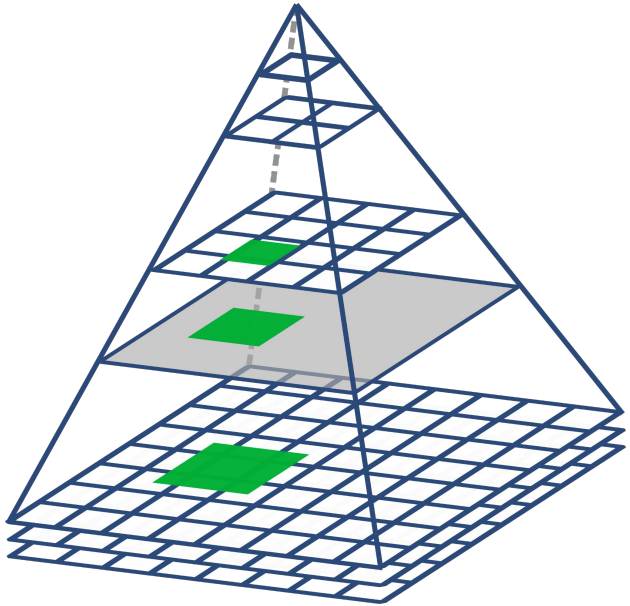
GoFunc

Deep Zoom Generator

Как получить все тайлы

- Получаем все уровни разрешения в исходном файле
- Вычисляем “виртуальные” уровни – Deep Zoom Levels и определяем количество тайлов на каждом из них

➔ Производим Downsampling – для каждого Deep Zoom Level определяем реальный уровень и коэффициент сжатия



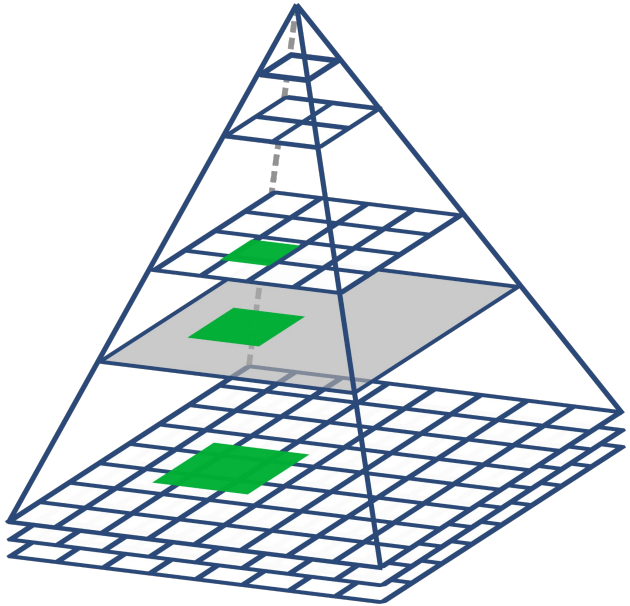
GoFunc

Deep Zoom Generator

Как получить все тайлы

- Получаем все уровни разрешения в исходном файле
- Вычисляем “виртуальные” уровни – Deep Zoom Levels и определяем количество тайлов на каждом из них
- Производим Downsampling – для каждого Deep Zoom Level определяем реальный уровень и коэффициент сжатия

➔ Запускаем обход каждого уровня и с помощью даунсемплинга получаем каждый тайл



GoFunc

Deep Zoom Generator


```go
func newDeepZoom( ... ) *deepZoom {
  // Получаем все уровни
  levels := getLevelDimensions(slide, limitBounds)

  // Количество уровней для DeepZoom
  deepZoomLevels := getDeepZoomLevels(levels[0])

  // Определяем количество тайлов по высоте и ширине
  tileLevels := getDeepZoomTileLevels(tileSize, deepZoomLevels)

  // маппим координаты с DeepZoom на реальные уровни
  downsamples := generateDownsamples(slide, tileLevels)

  return &deepZoom{ ... }
}
```

GoFunc

Profiling & Optimizations

Benchmark


```go
func BenchmarkReadTiles(b *testing.B) {
  tests := []struct{ ... }{
      ...
      {basename: "SOFT tissue vv-005.svs"},
  }
  for _, test := range tests {
      require.NoError(b, err, ... )
      b.Run( ... , func(b *testing.B) {
          err := readTilesWorker(ctx, t.basename)
          require.NoError(b, err)
      })
  }
}
```

GoFunc

Benchmark


```
BenchmarkReadTiles/CMU-1-Small-Region.svs_1.9_MB_th20-8
1000000000          0.5279 ns/op        1 B/op        0 allocs/op
PASS
ok     gitlab.histoint.net/histo/tools/openslidego/benchtestv2 17.164s
```

17.164s - He ok

GoFunc

Benchmark


```go
gen, err := openslidego.New( ... )
if err ≠ nil { ... }

jobsCh := gen.Iter(ctx)

g, gCtx := errgroup.WithContext(ctx)
for i := 0; i < threadsCount; i++ {
  g.Go(func() error {
     return readTilesWorkerPoolWorkerFunc(gCtx, gen, jobsCh)
  })
}
```

GoFunc

Benchmark


```
BenchmarkReadTiles/CMU-1-Small-Region.svs_1.9_MB_th20-8          1
1360151959 ns/op      1716105936 B/op    4242 allocs/op
PASS
ok    gitlab.histoint.net/histo/tools/openslidego/benchtestv2 1.661s
```

1.661s - на 128 коннектах ок?

GoFunc

TCMAlloc

Добавляем TCMAlloc

```
LD_PRELOAD=/lib/x86_64-linux-gnu/libtcmalloc.so.4
go test -bench=. -benchmem -timeout 30m -test.bench
BenchmarkTileGenerators ./benchtestv2
```

Profiling

Включаем pprof

```
import (
  _ "github.com/restream/reindexer/v3/pprof"
)
```

Запускаем с HEAPPROFILE

```
HEAPPROFILE=/tmp/pprof
go test -bench=. -benchmem -timeout 30m -test.bench
BenchmarkReadTiles ./benchtestv2
```

Снимаем профиль

```
pprof -symbolize remote … /debug/pprof/heap
```

GoFunc

Profiling

Сэтапим тест

- File: 1.7GB
- JPG 90%
- 100 goroutines
- Disable swap
- Drop page cache

GoFunc

Profiling

Сэтапим тест

- File: 1.7GB
- JPG 90%
- 100 goroutines
- Disable swap
- Drop page cache

Прогоняем на SSD – **~120s**

Profiling

Сэтапим тест

- File: 1.7GB
- JPG 90%
- 100 goroutines
- Disable swap
- Drop page cache

Прогоняем на SSD – **~120s**
Прогоняем на tmpfs – **~120s**

Profiling

Сэтапим тест

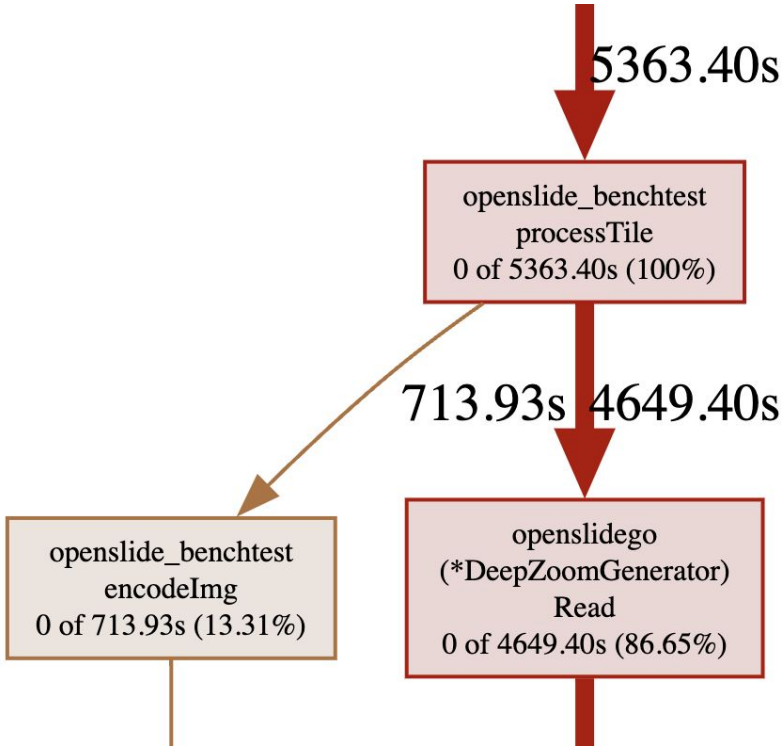
- File: 1.7GB
- JPG 90%
- 100 goroutines
- ~~Disable swap~~ – не надо так делать
- ~~Drop page cache~~ – и так тоже не стоит

GoFunc

Profiling

Упираемся не в I/O

- с много на преобразование в png/jpg/webp
- ReadRegion в OpenSlide не оптимиальный

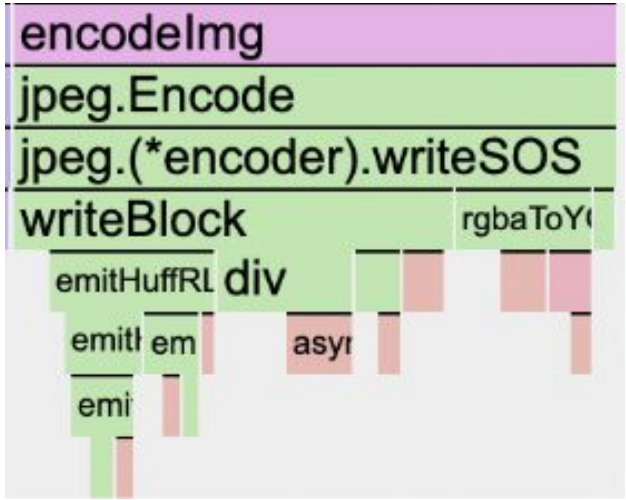


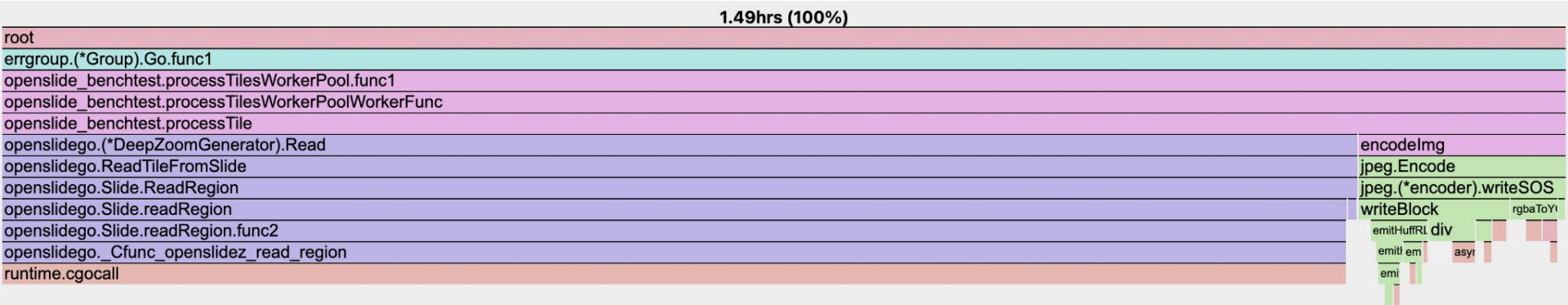
GoFunc

Profiling

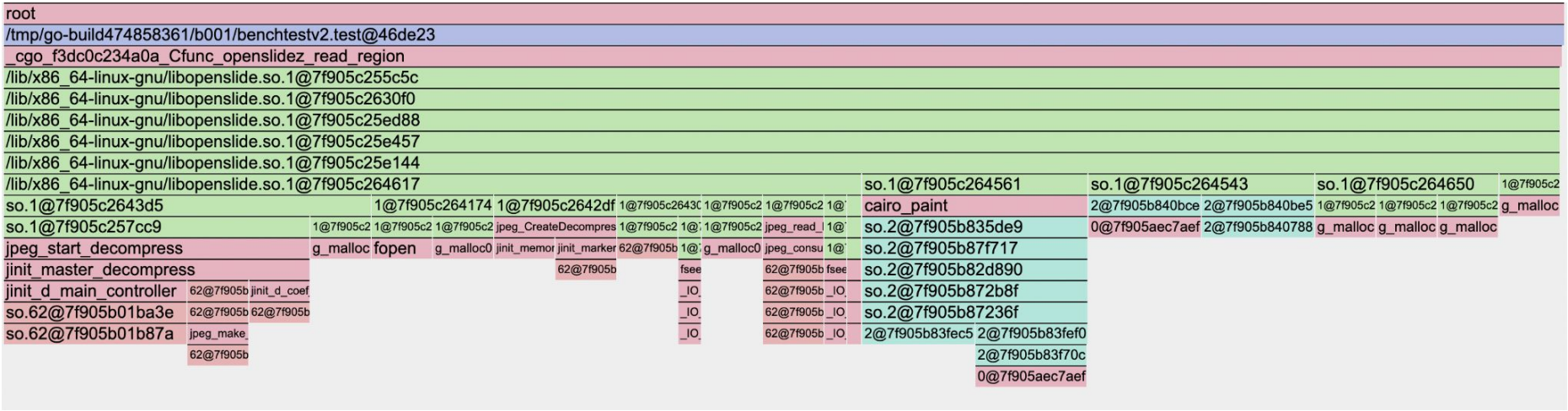
runtime
cgocall
4609.92s (85.92%)
of 4609.94s (85.92%)

openslide_benchtest.test



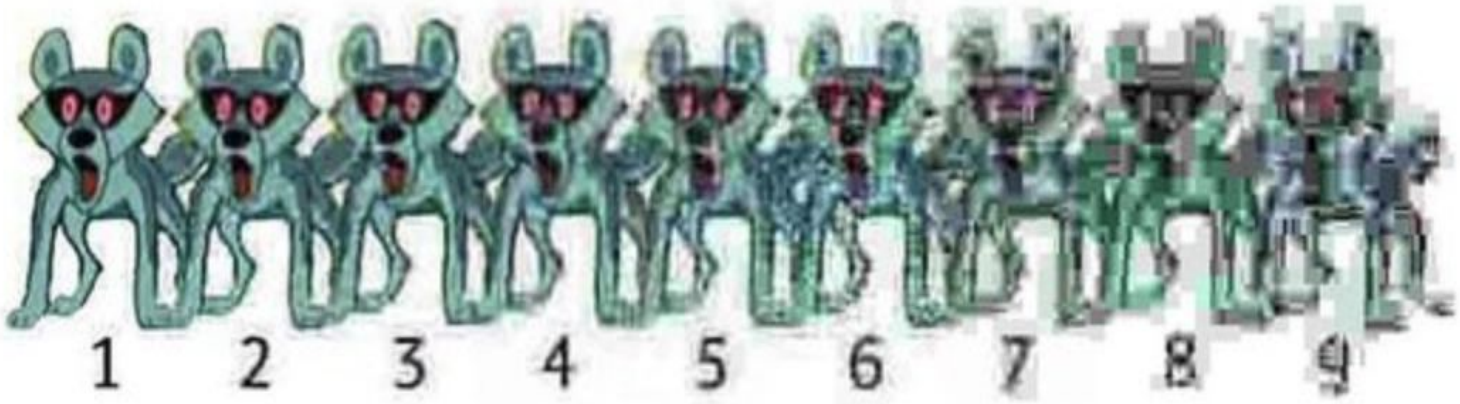


Profiling


root
/tmp/go-build474858361/b001/benchtestv2.test@46de23
_cgo_f3dc0c234a0a_Cfunc_openslidez_read_region
/lib/x86_64-linux-gnu/libopenslide.so.1@7f905c255c5c
/lib/x86_64-linux-gnu/libopenslide.so.1@7f905c2630f0
/lib/x86_64-linux-gnu/libopenslide.so.1@7f905c25ed88
/lib/x86_64-linux-gnu/libopenslide.so.1@7f905c25e457
/lib/x86_64-linux-gnu/libopenslide.so.1@7f905c25e144
/lib/x86_64-linux-gnu/libopenslide.so.1@7f905c264617
so.1@7f905c264561
so.1@7f905c264543
so.1@7f905c264650
so.1@7f905c2643d5
1@7f905c264174
1@7f905c2642df
cairo_paint
2@7f905b840bce
2@7f905b840be5
so.1@7f905c257cc9
so.2@7f905b835de9
0@7f905aec7aef
2@7f905b840788
g_malloc
jpeg_start_decompress
fopen
g_malloc0
jpeg_CreateDecompres
jpeg_read_
so.2@7f905b87f717
jpeg_consu
jinit_master_decompress
jinit_marker
62@7f905b
fsee
so.2@7f905b82d890
jinit_d_main_controller
jinit_d_coef
_IO
so.2@7f905b872b8f
so.62@7f905b01ba3e
so.2@7f905b87236f
so.62@7f905b01b87a
jpeg_make_
2@7f905b83fec5
2@7f905b83fef0
2@7f905b83f70c
0@7f905aec7aef

Вместо заключения

Эксперименты



Сорри, но сейчас будут шакалы ...

GoFunc

Эксперименты

Artem Prikhodin 2:21 PM

Новый мировой рекорд процессинга - 4гб слайд за 56 секунд

image.png

```
ystem 3070% cpu 56.601 total
rget/release/process 50 /mnt/
```

Artem Prikhodin 5:23 PM

Прошлый мировой рекорд отменяется, новый теперь 48 сек на 4гб слайд (edited)

image.png

```
cess 50   1458.89s user 32.82s system 3076
```

Artem Prikhodin 5:41 PM

3.1 гб ndpi с демо2 с 64к тайлами за 31 секунду

image.png

```
m 3025% cpu 31.302 total
mnt/archive/slides/demo2/2023/10/05/a83bbfd
5a1-4a45-a0a6-e94f47d07b7f/2023-10-04 11.39
```

GoFunc

Выводы

Что получили

- Достигли стабильных ~90s для подготовки изображения
- Отскейлили процессинг для большего количества файлов

Выводы

Что получили

- Достигли стабильных ~90s для подготовки изображения
- Отскейлили процессинг для большего количества файлов

Вопросы и гипотезы

- Rust быстрее, но сложнее 🤯
- Переход с CPU на GPU?
- Как отказаться от Openslide

Пожалуйста, оставьте фидбэк

Евгений Конечный
Uzum Tezkor