openslide 安装

官方地址
openslide-python 安装先决条件

```
Python 3 >= 3.6
```

macos 安装

```
brew install opencv
brew install openslide
pip install openslide-python
```

openslide图像切割

官方支持医学病理图片格式

- .svs
  主要应用有医学病理图像
- .tif,tiff
  Tagged Image File Format。
  该格式支持256色、24位真彩色、32位色、48位色等多种色彩位，同时支持RGB、CMYK及YCbCr等多种色彩模式。
  TIFF文件可以是不压缩的，文件较大，也可以是压缩的，支持RAW、RLE、LZW、JPEG、CCITT3组和4组等多种压缩方式。
- .vms
  Virtual Microscope Specimen 虚拟显微镜标本
- .vmu
  Uncompressed Virtual Microscope Specimen 未压缩的虚拟显微镜标本
- .ndpi
  NanoZoomer Digital Pathology Image 纳米缩放数字病理图像
- .scn
- .mrxs
- .svslide
- .bif

其他测试后支持的图片格式

- jpg
- jpeg

- png
- bmp

处理步骤

1.导入openslide，通过传入图像路径imagePath，打开图像，返回OpenSlide对象。

```
import openslide
slide = openslide.open_slide(imagePath)
```

open_slide函数打开方式，当filename为 whole-slide 图像时（svs、tif等格式），返回OpenSlide对象，当为其他格式图像时（png、jpg等），返回ImageSlide对象

2.导入DeepZoomGenerator模块，实现从slide对象生成单个 Deep Zoom 切片的功能。

```
from openslide.deepzoom import DeepZoomGenerator
zoomslide = openslide.deepzoom.DeepZoomGenerator(osr=slide, tile_size=1024, overla
p=0, limit_bounds=False)
```

- osr: openslide打开图像生成的slide对象
- tile_size: 单个瓦片的宽度和高度，设置切割图像的大小，必须遵守的规则是 `tile_size + 2 * overlap` 是2的幂次，这样可以获得更好的展示性能。
- overlap: 添加到图块的每个内部边缘的额外像素数，指图像切割时，是否有overlap个像素重叠。
- limit_bounds: 若为True，表示仅渲染非空区域，表示的是大图整个边缘可能达不到自己设的长和宽。False,则丢弃边缘图。True,保存。

原始slide图像根据不同分辨率分成多层，并在不同分辨率下切割成多个块。

3.保存切割的tiles图像和生成dzi文件

```
#保存tiles图片
def write_tiles(deepzoomslide, format, basename):
    for level in range(deepzoomslide.level_count):
        tiledir = os.path.join(tilePath, "%s_files" % basename, str(level))
        if not os.path.exists(tiledir):
            os.makedirs(tiledir)
        cols, rows = deepzoomslide.level_tiles[level]
        for row in range(rows):
            for col in range(cols):
                tilename = os.path.join(tiledir,'%d_%d.%s' %(col,row,format))
                if not os.path.exists(tilename):
                    tile = deepzoomslide.get_tile(level, (col, row))
                    tile.save(tilename)
write_tiles(zoomslide, format, basename)

#保存dzi文件
def write_dzi(deepzoomslide, format, basename):
    dziPath = tilePath + "%s.dzi" % basename
    with open(dziPath, 'w') as f:
        f.write(deepzoomslide.get_dzi(format))

write_dzi(zoomslide, format, basename)
```

4.前端展示

前端展示需要借用openseadragon

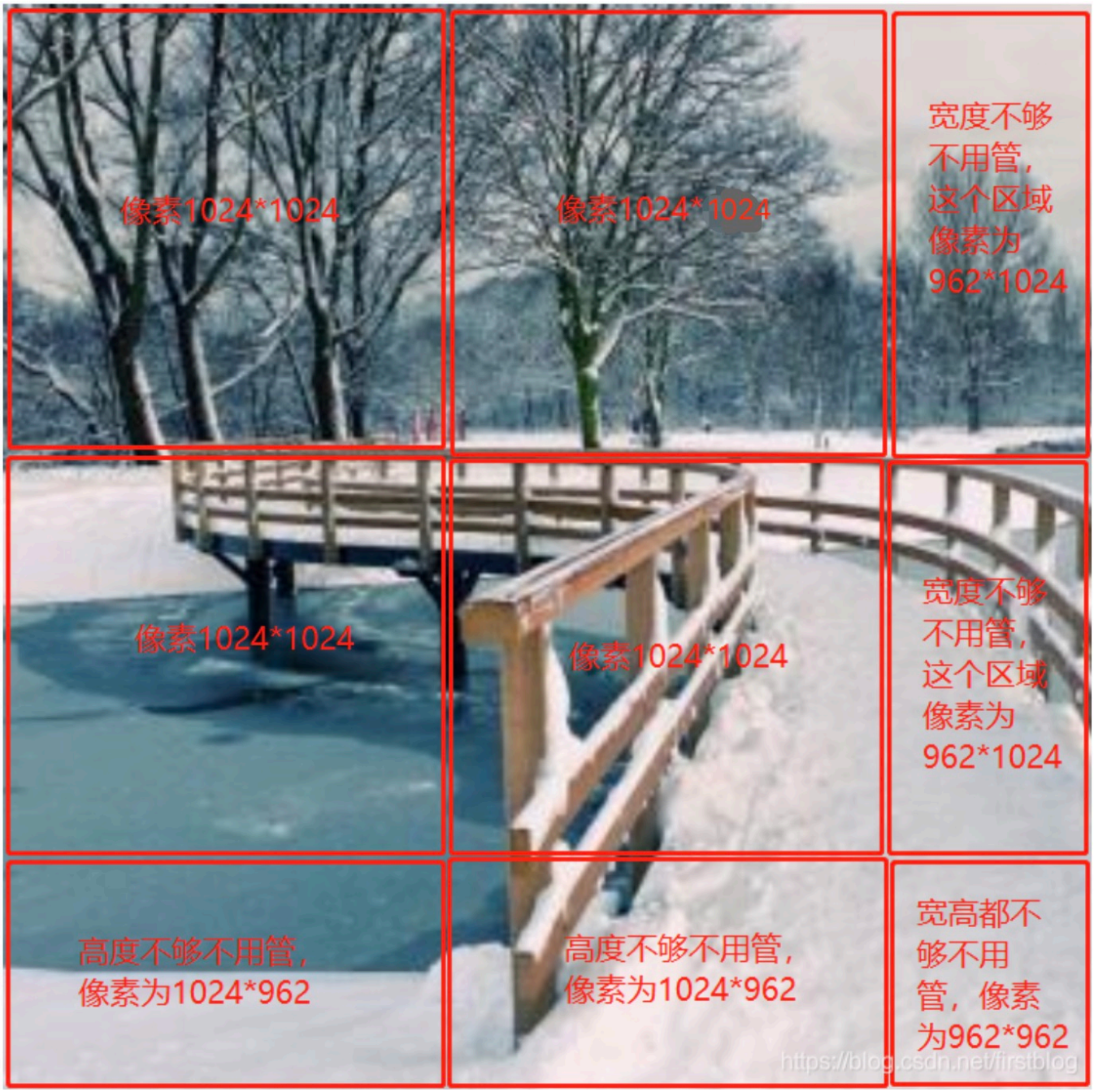
[openseadragon官方地址]

```
<div id="openseadragon1" style="width: 800px; height: 600px;"></div>
    <script src="./openseadragon-bin-3.0.0/openseadragon.min.js"></script>
    <script type="text/javascript">
        var viewer = OpenSeadragon({
            id: "openseadragon1",//绑定ID
            //openseadragon自带的一些图标
            prefixUrl: "./openseadragon-bin-3.0.0/images/",
            tileSources: {
                Image: {
                    xmlns: "http://schemas.microsoft.com/deepzoom/2009",//默认
                    Url: "./tiles_files/",//分割后tiles图片存放的路径
                    Overlap: "0",//图像分割时指定的参数
                    TileSize: "1024",
                    Format : "jpg",
                    Size:{
                        Height: "21952",//原图维度信息
                        Width:  "30000"
                    }
                }
            }
        });
    </script>
```

切割规则

图像切割规则

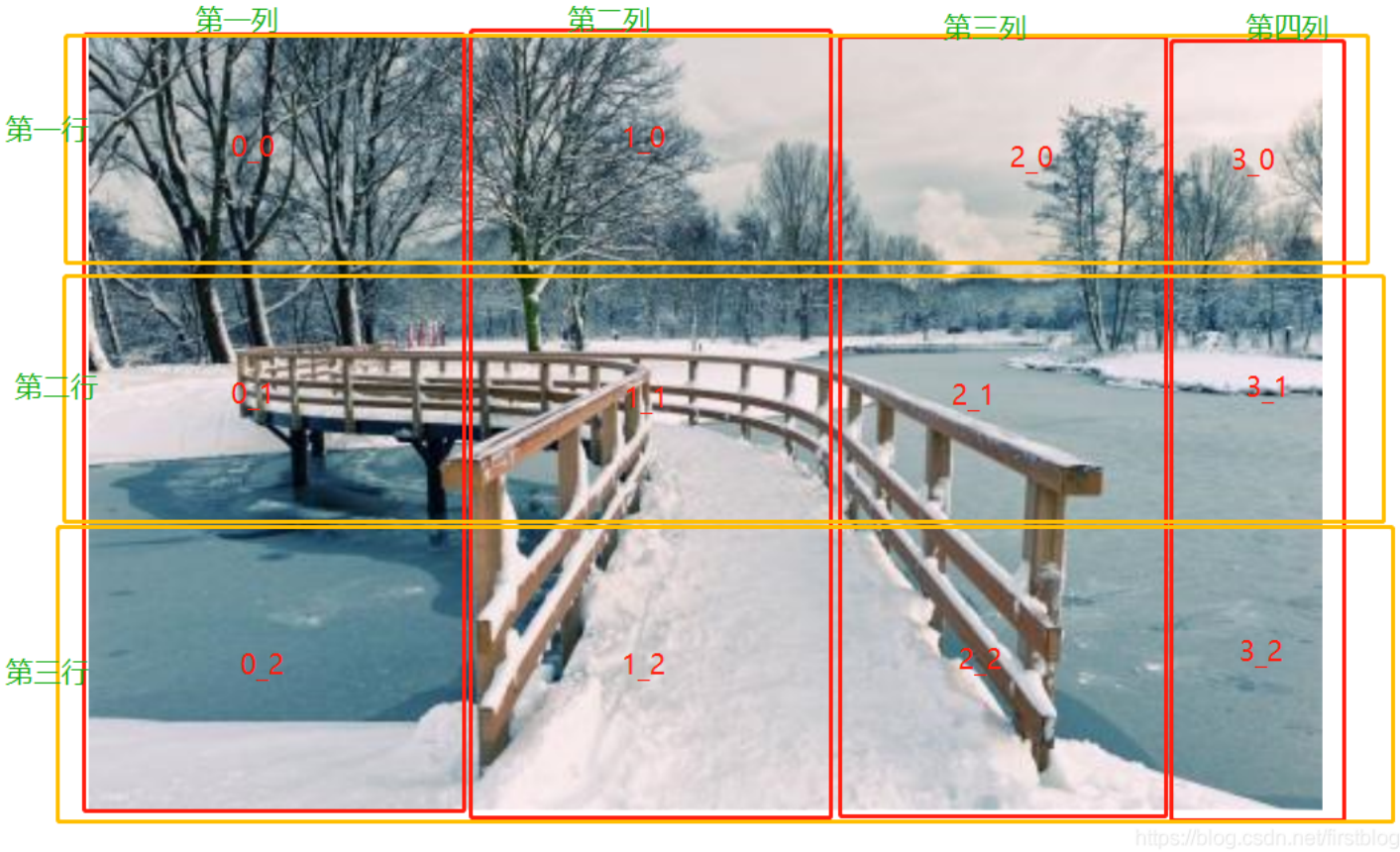
- 1.切割图像之前需要定义切割后每一张图片的大小为多少像素，每张切割出来的图像必须为正方形，且有一个规则，即必须是2的N次方。若图像边缘不够，无需额外操作，举个例子，tile_size指定为1024时：



- 2.原图像一定是有多张且像素比不同的图像（即不同层级level_count）（这个可以通过对原图像进行维度缩放实现），以便更加清晰的显示图像。每种像素的图片，都应该按照同样的规则进行切割，即切割大小须一致。

图像命名规则

图像切割后，只需要按照一定的规则放置图片，openseadragon会自动去相应目录读取文件。
每一张切割的图片都需要进行保存，保存的命名格式为 `col_row.format` ，col和row都是从0开始，format为jpg或png。
命名规则示例：



相同层级切割的图像放在同一个目录下，不同层级的放在不同的目录，目录的命令规则如下：
在哪一个层级，目录的命名就是层级序号，从0开始。

这些图像切割后的目录放在一个上级目录下，与dzi文件在同级目录。命名方式与dzi文件的命名有关，例如test.dzi，目录的命名即为test_files。
以JP2K demo为例：

```
JP2K
├── tiles.dzi
└── tiles_files
    ├── 0
    ├── 1
    ├── 10
    ├── 11
    ├── 12
    ├── 13
    ├── 14
    ├── 15
    ├── 16
    ├── 2
    ├── 3
    ├── 4
    ├── 5
    ├── 6
    ├── 7
    ├── 8
    └── 9
```

DZI图像规则官方文档

图像处理时的一些指标信息

- 不同图像大小，指定相同tile_size

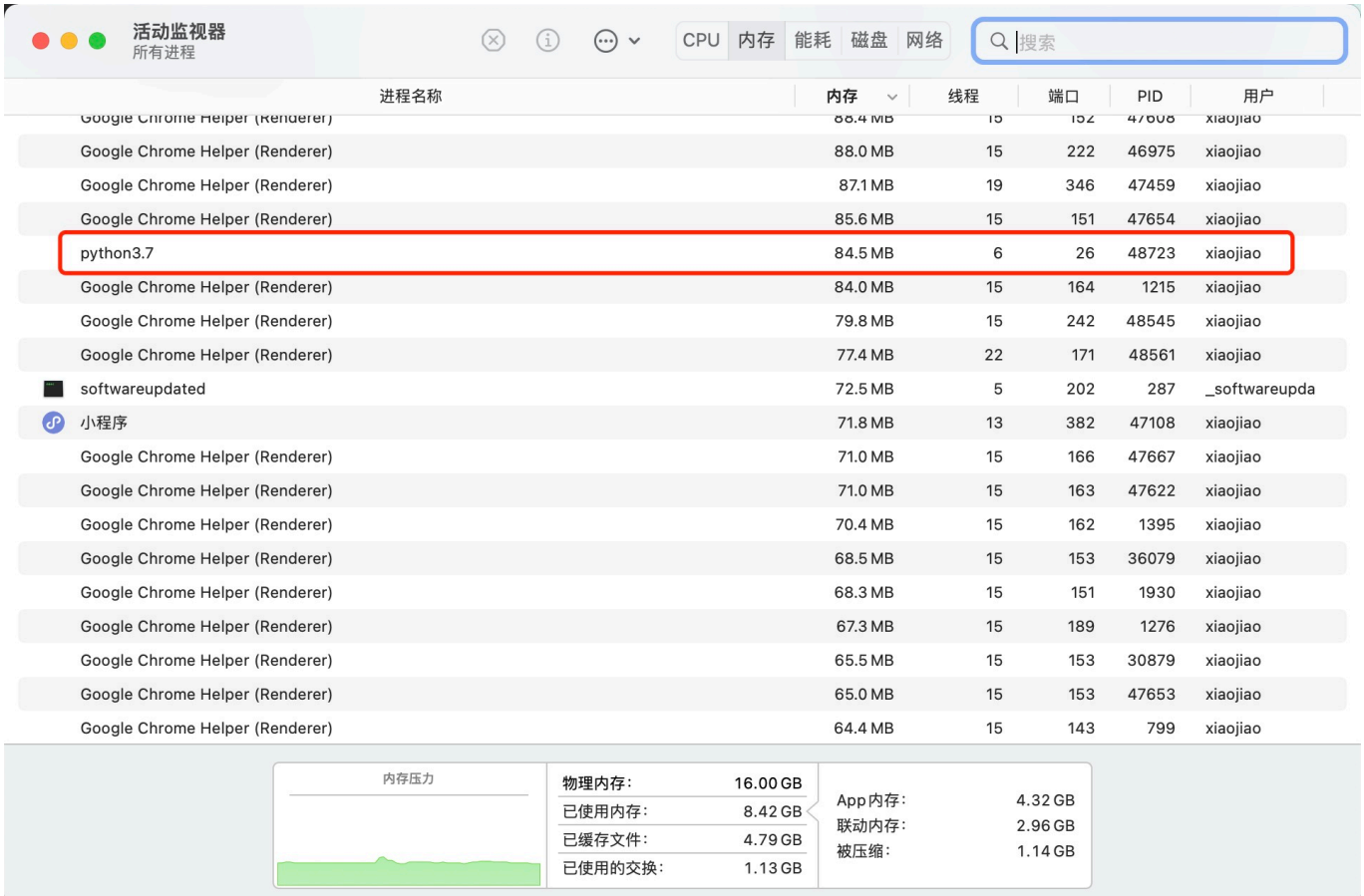
图像名称	原图大小	原图分辨率	tile_size	生成层数	tiles总数	处理时间
The Tower of Babel.jpeg	202.82MB	(30000,21952)	256	16	13583	244s
Summer Evening at Skagen beach.jpg	145.8MB	(16403,11698)	256	16	4039	49s
JP2K-33003-2.svs	275.85MB	((32671, 47076), (8167, 11769), (2041, 2942))	256	17	31417	397s

处理过程中机器的指标：

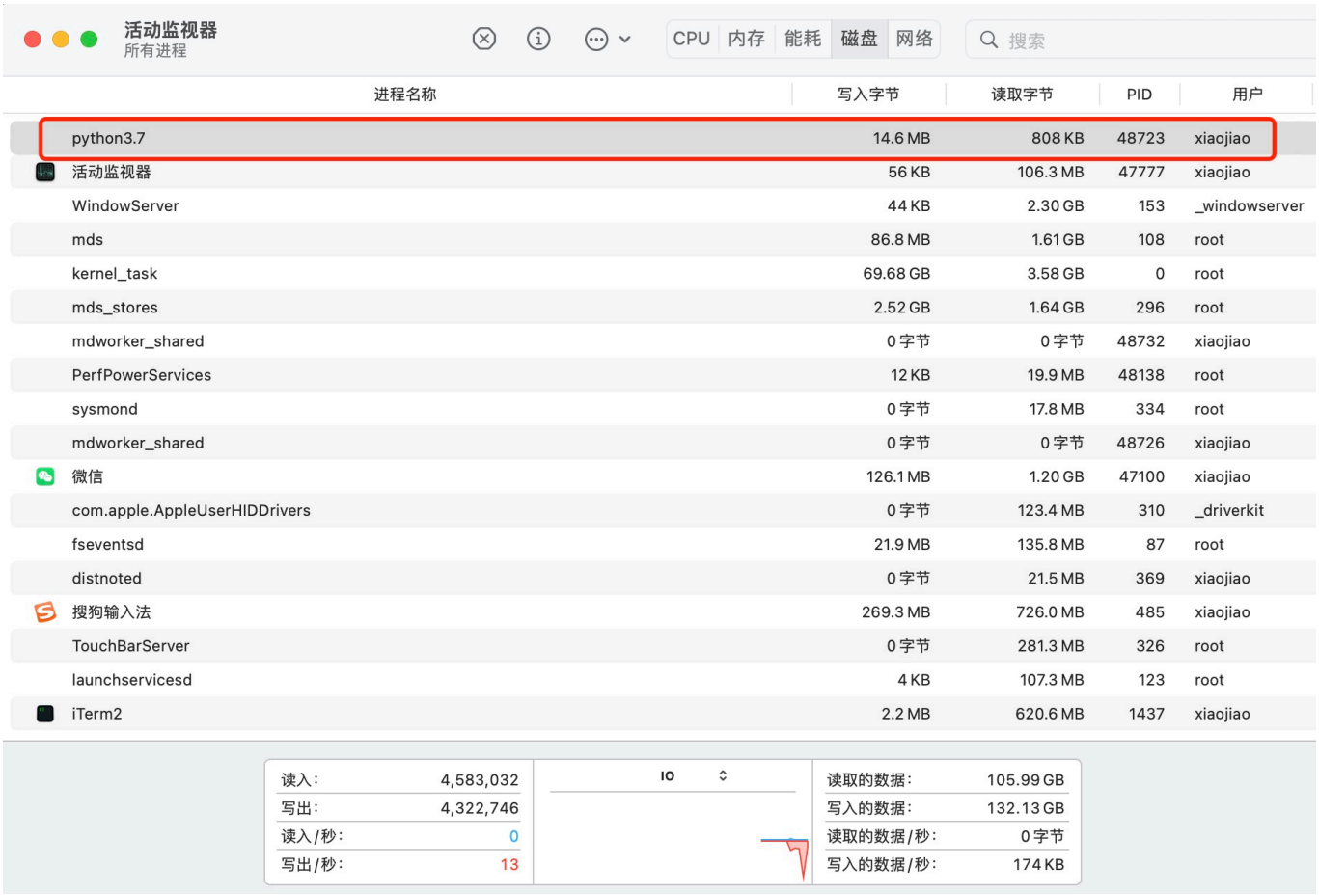
- CPU指标


活动监视器
所有进程
CPU 内存 能耗 磁盘 网络
搜索
进程名称 % CPU CPU时间 线程 闲置唤醒 % GPU GPU时间 PID 用户
python3.7 99.8 19.39 6 0 0.0 0.00 48723 xiaojiao
WindowServer 9.9 5:45:50.21 16 45 2.5 8:55:35.31 153 _windowserver
活动监视器 0.00 47777 xiaojiao
kernel_task 0.00 0 root
mds 0.00 108 root
mds_stores 0.00 296 root
com.apple.A 0.00 310 _driverkit
mdworker_s 0.00 48726 xiaojiao
微信 0.00 47100 xiaojiao
sysmond 0.00 334 root
iTerm2 0.42 1437 xiaojiao
launchd 0.00 1 root
TouchBarSe 3:26.98 326 root
fseventsd 0.00 87 root
Google Chr 0.24 767 xiaojiao
mdworker_s 0.00 48721 xiaojiao
launchservi 0.00 123 root
访达 0.1 40:06.64 4 0 0.0 0.11 383 xiaojiao
CPU历史记录
内核1
内核2
内核3
内核4
内核5
内核6
内核7
内核8
内核9
内核10
内核11
内核12
系统: 1.07%
用户: 9.43%
闲置: 89.50%
CPU负载
线程: 2,274
进程: 530

- 内存指标


活动监视器
所有进程
CPU 内存 能耗 磁盘 网络
搜索
进程名称 内存 线程 端口 PID 用户
Google Chrome Helper (Renderer) 88.0 MB 15 222 46975 xiaojiao
Google Chrome Helper (Renderer) 87.1 MB 19 346 47459 xiaojiao
Google Chrome Helper (Renderer) 85.6 MB 15 151 47654 xiaojiao
python3.7 84.5 MB 6 26 48723 xiaojiao
Google Chrome Helper (Renderer) 84.0 MB 15 164 1215 xiaojiao
Google Chrome Helper (Renderer) 79.8 MB 15 242 48545 xiaojiao
Google Chrome Helper (Renderer) 77.4 MB 22 171 48561 xiaojiao
softwareupdated 72.5 MB 5 202 287 _softwareupda
小程序 71.8 MB 13 382 47108 xiaojiao
Google Chrome Helper (Renderer) 71.0 MB 15 166 47667 xiaojiao
Google Chrome Helper (Renderer) 71.0 MB 15 163 47622 xiaojiao
Google Chrome Helper (Renderer) 70.4 MB 15 162 1395 xiaojiao
Google Chrome Helper (Renderer) 68.5 MB 15 153 36079 xiaojiao
Google Chrome Helper (Renderer) 68.3 MB 15 151 1930 xiaojiao
Google Chrome Helper (Renderer) 67.3 MB 15 189 1276 xiaojiao
Google Chrome Helper (Renderer) 65.5 MB 15 153 30879 xiaojiao
Google Chrome Helper (Renderer) 65.0 MB 15 153 47653 xiaojiao
Google Chrome Helper (Renderer) 64.4 MB 15 143 799 xiaojiao
内存压力
物理内存: 16.00 GB
已使用内存: 8.42 GB
已缓存文件: 4.79 GB
已使用的交换: 1.13 GB
App内存: 4.32 GB
联动内存: 2.96 GB
被压缩: 1.14 GB

- 磁盘指标



tile*size指定为256时，主要的处理瓶颈是CPU - 相同图像，指定不同tile*size

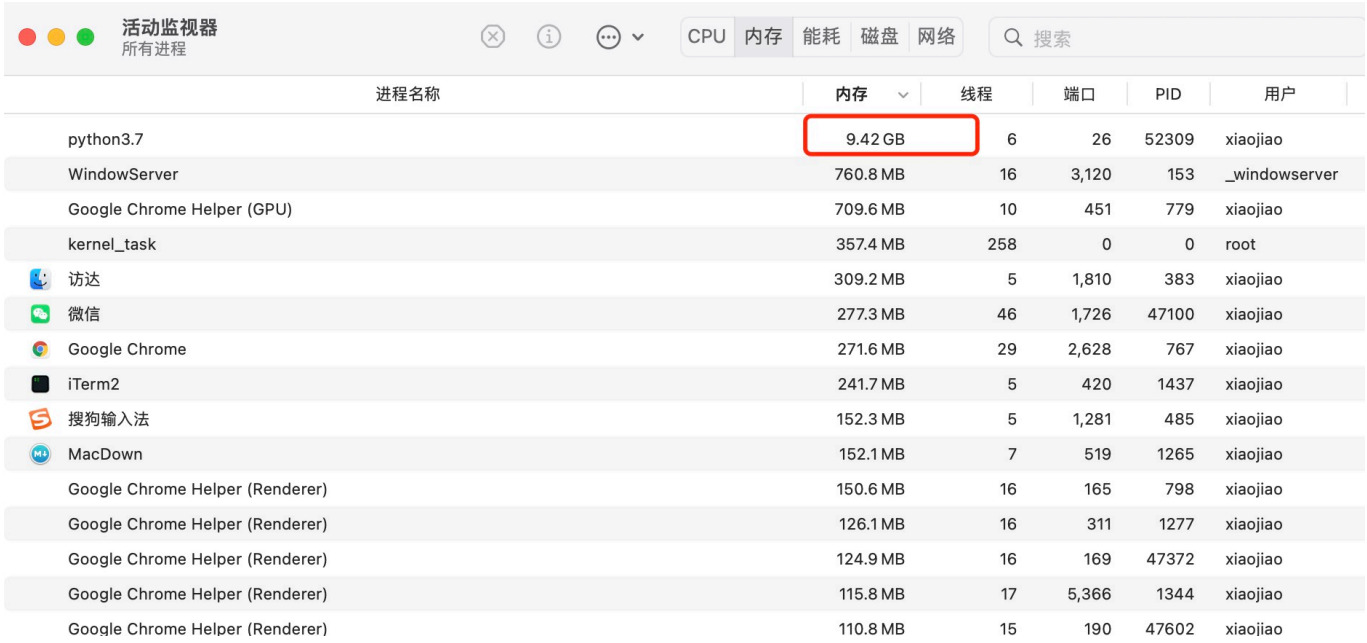
图像名称	原图大小	原图分辨率	tile_size	生成层数	tiles总数	处理时间
The Tower of Babel.jpeg	202.82MB	(30000,21952)	256	16	13583	244s
The Tower of Babel.jpeg	202.82MB	(30000,21952)	512	16	3436	188s
The Tower of Babel.jpeg	202.82MB	(30000,21952)	1024	16	900	191s

处理过程中机器的指标：

- CPU指标

进程名称	% CPU	CPU时间	线程	闲置唤醒	% GPU	GPU时间	PID	用户
python3.7	99.8	2:12.24	6	0	0.0	0.00	52309	xiaojiao
WindowServer	7.1	6:45:53.56	15	22	0.0	9:29:53.97	153	_windowserver
活动监视器	5.2	23.76	7	6	0.0	0.00	52170	xiaojiao
微信	3.0	29:22.50	43	24	0.0	0.00	47100	xiaojiao
kernel_task	1.9	2:33:01.85	258	461	0.0	0.00	0	root
com.apple.AppleUserHIDDriv...	1.2	19:04.81	3	0	0.0	0.00	310	driverkit

- 内存指标



进程名称	内存	线程	端口	PID	用户
python3.7	9.42 GB	6	26	52309	xiaojiao
WindowServer	760.8 MB	16	3,120	153	_windowserver
Google Chrome Helper (GPU)	709.6 MB	10	451	779	xiaojiao
kernel_task	357.4 MB	258	0	0	root
访达	309.2 MB	5	1,810	383	xiaojiao
微信	277.3 MB	46	1,726	47100	xiaojiao
Google Chrome	271.6 MB	29	2,628	767	xiaojiao
iTerm2	241.7 MB	5	420	1437	xiaojiao
搜狗输入法	152.3 MB	5	1,281	485	xiaojiao
MacDown	152.1 MB	7	519	1265	xiaojiao
Google Chrome Helper (Renderer)	150.6 MB	16	165	798	xiaojiao
Google Chrome Helper (Renderer)	126.1 MB	16	311	1277	xiaojiao
Google Chrome Helper (Renderer)	124.9 MB	16	169	47372	xiaojiao
Google Chrome Helper (Renderer)	115.8 MB	17	5,366	1344	xiaojiao
Google Chrome Helper (Renderer)	110.8 MB	15	190	47602	xiaojiao

- 磁盘指标

进程名称	写入字节	读取字节	PID	用户
python3.7	44 KB	617.7 MB	52309	xiaojiao
活动监视器	40 KB	57.9 MB	52170	xiaojiao
WindowServer	44 KB	3.02 GB	153	_windowserver
sysmond	0 字节	26.0 MB	334	root
kernel_task	94.71 GB	5.23 GB	0	root
PerfPowerServices	4.0 MB	94.9 MB	48138	root
微信	299.9 MB	2.57 GB	47100	xiaojiao

tile_size指定大于256时，处理内存占用迅速增加，占用60%左右，主要的处理瓶颈依然是CPU，但占用内存增大也会影响处理速度。