Parallel Processing Images in Python

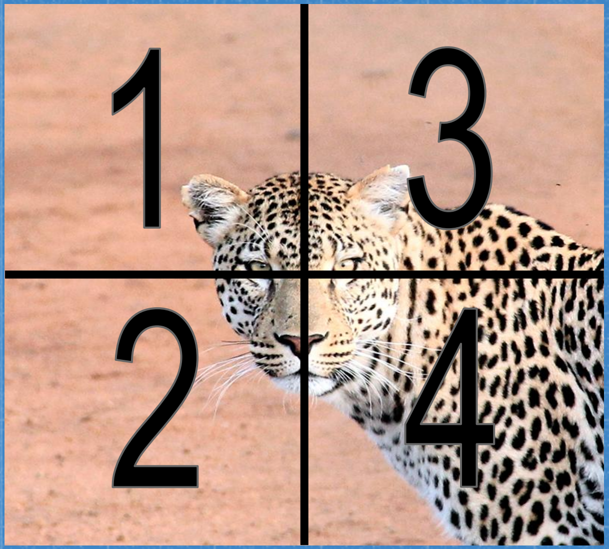
By Alisha G., Taylor K., Mark W., Seth S.

The Plan

- Parallelize image processing in Python
- Included parallelizing modules are multiprocessing and Threading
- Included image modules and methods are PIL and CV2
- Additional modules are time and numpy(for computations)
- Included methods are grayscale serial, threading, and multiprocessing
- Parallelizing methods are quadrant and height

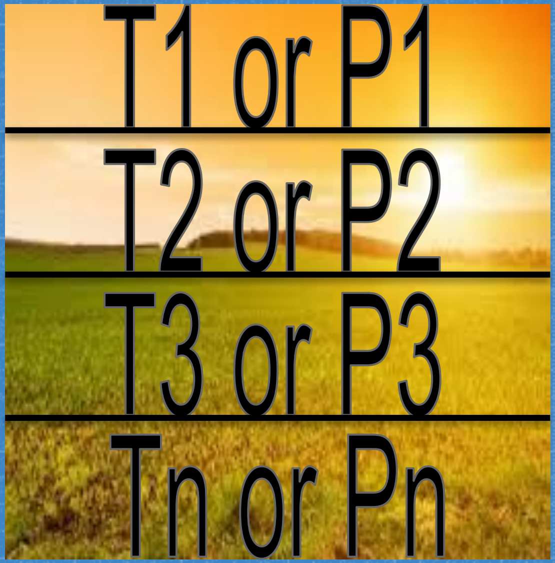
Quadrant Method

- Image is separated into 4 quadrants and processed according (processing order can be manipulated)



Height Method

- Image is separated by height according to user’s preference and processed top down.
- Min being one separation and Max being the height dimension.



PIL Method

- PIL was released in 1995 as one of Python’s first image processing libraries
- It is now more commonly known as Pillow (pip install pillow)
- The Commands used for image manipulation:
    - Img = PIL.Image.open(fp, mode='r')
    - W,H = PIL.Image.size
    - New_img = PIL.Image.new(mode, size, color=0)
    - Converted_img =Image.convert(mode=None, matrix=None, dither=None, palette=0, colors=256)
    - R,G,B = Image.getpixel((x,y))
    - Image.putpixel(xy, value)
    - Image.show(title=None, command=None)

CV2 Method

- OpenCV (Open Source Computer Vision), developed in 2000 by intel.
- OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations.
- The commands used for image manipulation:
  - Img = cv2.imread(‘img_path’)
  - H,W = cv2.shape[:2]
  - New_img = np.zeros((h,w),np.uint8)
  - New_img[y,x] = np.clip(0.07 * img[y,x,0] + 0.72 * img[y,x,1] + 0.21 * img[y,x,2], 0, 255)
  - cv2.imshow(‘window_name’,New_img)
  - cv2.waitKey(0)

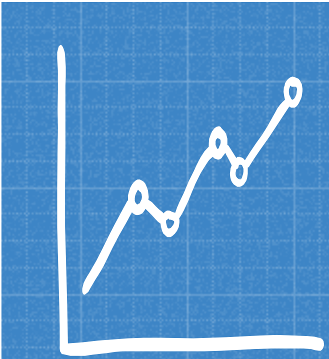
Serial Results

CV2 Grayscale Time vs image size

600x600	900x900	1200x1200
2 secs	6 secs	11 secs

PIL Grayscale Time vs image size

200x200	400x400	600x600
0 secs	43 secs	222 secs



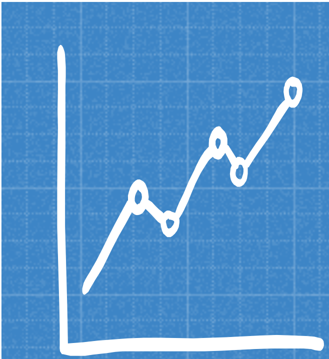
PIL Results

Grayscale Time vs image size(Quadrant Method)

Type	200x200	400x400	600x600
Thread	2 secs	30 secs	134 secs
Multiprocess	**2 secs**	**44 secs**	**226 secs**

Grayscale Time vs # Processes (Height Method 400x400)

Type	4 processes	8 processes	16 processes
Thread	2 secs	28 secs	28 secs
Multiprocess	**47 secs**	**50 secs**	**50 secs**



CV2 Results

Grayscale Time vs image size(Quadrant Method)

Type	600x600	900x900	1200x1200
Thread	3 secs	7 secs	12 secs
Multiprocess	**4 secs**	**7 secs**	**13 secs**

Grayscale Time vs # Processes (Height Method 900x900)

Type	4 processes	16 processes	256 processes
Thread	7 secs	7 secs	9 secs
Multiprocess	**2 secs**	**6 secs**	**87 secs**

CONCLUSIONS

- CV2 is significantly faster than PIL since it holds advance statistical libraries making it easy to manipulate pixel arrays
- The Quadrant method and the Height method roughly go the same amount of time with 4 processes.
- The threads go faster than the multi processes since the threads share memory
- As the image size increases so does the processing time

Demonstration