Distributed Object Detection

Universidade de Aveiro

Licenciatura em Engenharia Informática

UC 40382 - Computação Distribuída

Docentes:

Prof. Diogo Gomes

Prof. Nuno Lau

Trabalho realizado por:

Eduardo Santos - 93107

Pedro Bastos - 93150

Bibliotecas utilizadas

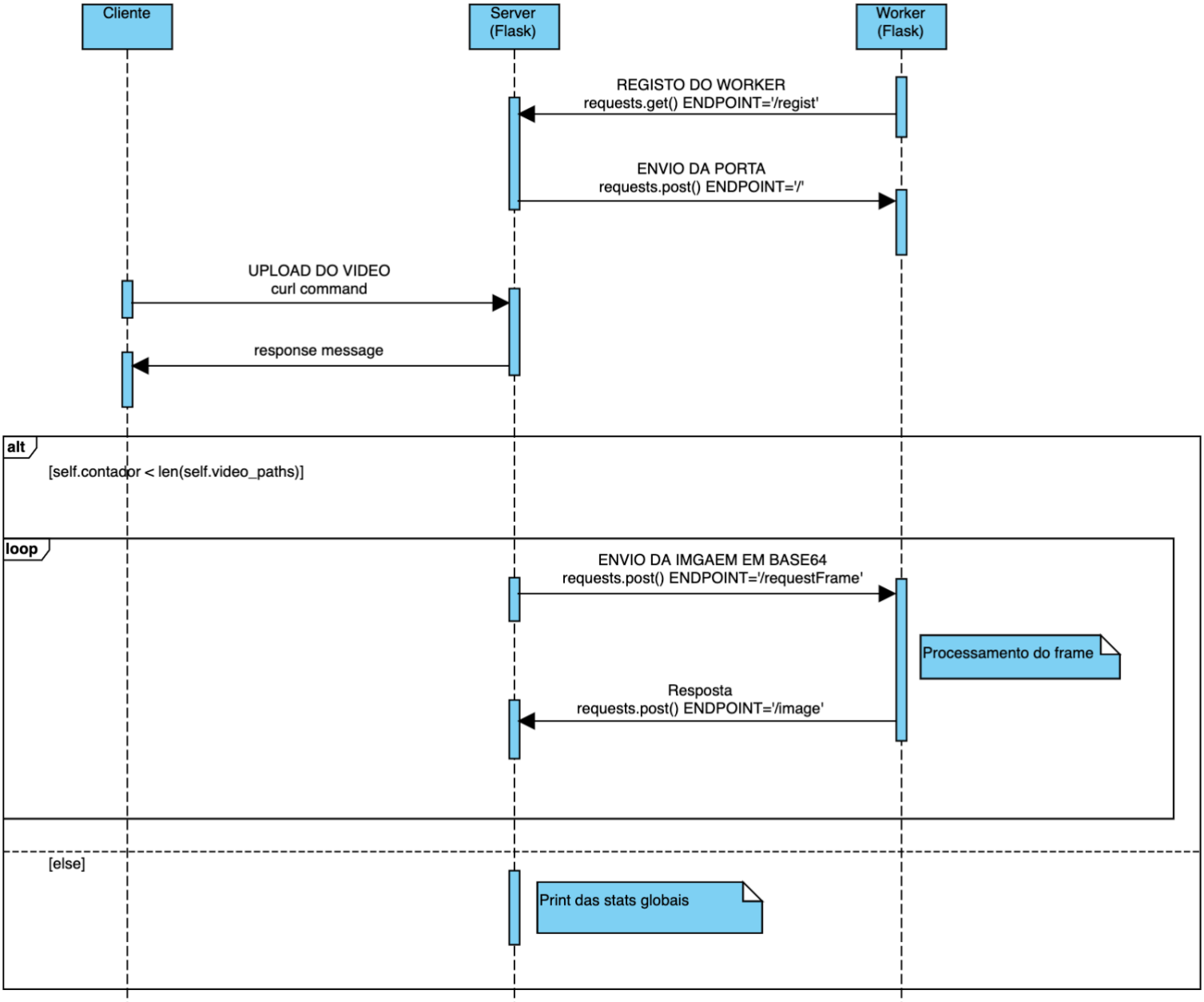
Tanto o server como os workers utilizam ***Flask***, uma framework de **Python** utilizada em Web. É utilizado um **ambiente virtual** para trabalhar localmente. Para os pedidos entre o server e o worker, é utilizada a biblioteca ***Python Requests***, que permite a interação entre os servidores **HTTP**. Finalmente, para poder enviar vários frames ao mesmo tempo para vários workers, utilizamos a biblioteca ***Python Asyncio*** que, além de simples de usar, é compatível com ***Flask*** e ***Python Requests***.

Protocolo

Primeiramente, é necessário iniciar o server, este que inicia o ***Flask*** e fica bloqueado à espera de pedidos. Em seguida, devem ser iniciados os workers pretendidos. Cada worker, depois de serem iniciadas as bibliotecas do ***tensorflow***, ***Keras*** e ***YOLOv3*** por uma questão de desempenho no processamento das frames, faz um pedido get ao URL do server. O servidor responde com a porta designada para esse worker. Depois do worker receber a porta, inicia o ***Flask*** com a mesma.

Seguidamente é feito o upload do vídeo do cliente para o servidor (“curl -F 'video=@moliceiro.m4v' http://localhost:5000”). O servidor recebe o video e responde com “Video delivered.”, não obrigando ao cliente esperar pelo processamento do video.

Depois de ler os frames todos do video e guardá-los numa lista, o servidor vai enviar **n** frames fazendo requests.post() de **n** em **n, sendo n o número de workers,** e espera pela resposta. Depois da resposta, verifica se o número de pessoas ultrapassa o definido e emite um alerta, caso existam razões para tal. Seguidamente, verifica se já estão todos os frames processados. Se não estiverem, envia os próximos **n** frames, se estiverem, é chamada a função ***endProcessing()*** que imprime na consola as estatísticas finais do vídeo.



Message sequence chart do protocolo

Resultados

Para a apresentação dos resultados, são impressos alertas no server quando, para cada frame, o número de pessoas presentes no mesmo ultrapassa o número definido inicialmente pelo utilizador. No mesmo alerta podemos ver o número do frame, o número de pessoas detectadas e a classe da mesmas (<person>):

```
Frame 420: 15 <person> detected
127.0.0.1 - - [25/Jun/2020 17:26:05] "POST /image HTTP/1.1" 200 -
```

Imagem: Exemplo de alerta

1 Worker:

```
Processed frames: 430
Average processing time per frame: 333ms
Person objects detected: 8540
Total classes detected: 6
Top 3 objects detected: person, boat, car

Tempo de execução do video: 0:38:46
```

2 Workers:

```
Processed frames: 431
Average processing time per frame: 658ms
Person objects detected: 8540
Total classes detected: 6
Top 3 objects detected: person, boat, car

Tempo de execução do video: 0:20:41
```

4 Workers:

```
Processed frames: 431
Average processing time per frame: 1240ms
Person objects detected: 8537
Total classes detected: 6
Top 3 objects detected: person, boat, car

Tempo de execução do video: 0:11:09
```

Referências

https://www.geeksforgeeks.org/get-post-requests-using-python/
https://docs.python.org/3/
https://requests.readthedocs.io/en/master/
https://docs.python.org/3/library/asyncio.html