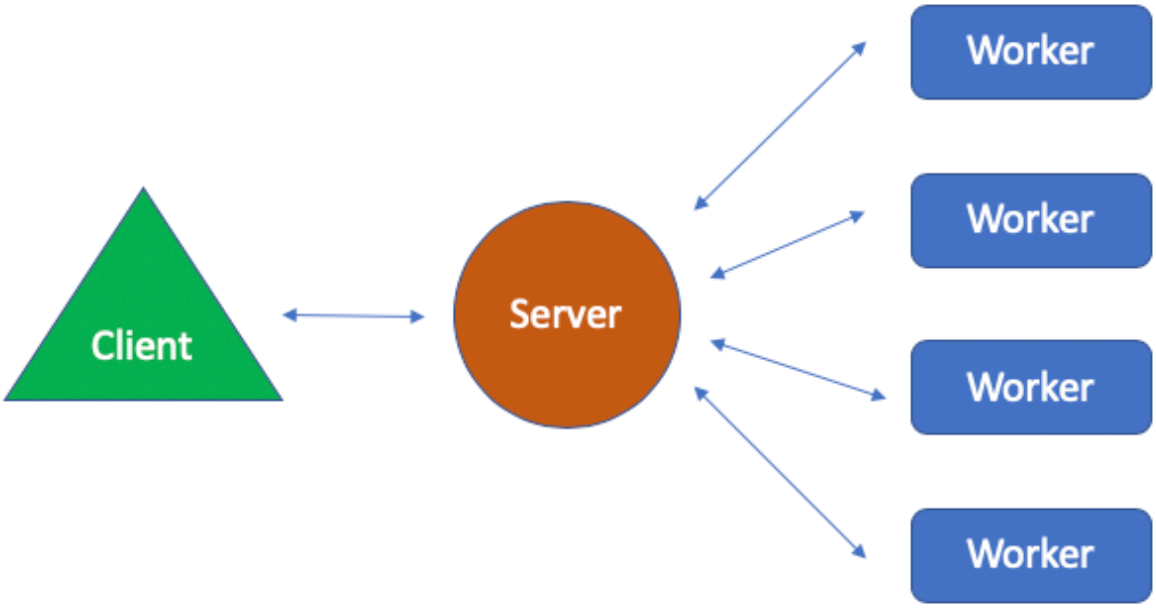
Projecto 2

Licenciatura em Engenharia Informática - Computação Distribuída

Distributed Object Detection

Docentes:
- Diogo Gomes (dgomes@ua.pt)
- Nuno Lau (nunolau@ua.pt)

Prazo:
25 de Junho – 24h00



Conceitos a abordar:

- Middleware's (HTTP)
- Marshalling
- Redundancy
- Fault Tolerance

Introdução

Nos tempos em que vivemos é essencial mantermos a distância social. As entidades de segurança realizam nestes tempos actividades de monitorização de ajuntamentos de pessoas, especialmente em ambiente turístico. Por forma a auxiliar esta tarefa, pretende-se desenvolver uma aplicação distribuída capaz de detectar pessoas num vídeo previamente gravado.

O cliente deve enviar um vídeo em formato MP4 para um servidor central através de protocolo HTTP, sem aguardar qualquer resposta do processamento final do servidor. Do servidor é esperado

que analise o vídeo e emita uma mensagem para a consola (do servidor) sempre que o número de pessoas num *frame* do video ultrapassou um dado limite fornecido por linha de comandos.

Para esta tarefa é fornecido este guião e um código seminal disponível em https://github.com/detiuaveiro/CD_distributed_object_detection.

Objectivos

Este é um trabalho aberto, em que se procura uma solução final que cumpra com o requisito de detectar se no vídeo existe em algum momento um número de pessoas superior a um valor pré-configurado.

Deve começar o trabalho por desenvolver uma solução centralizada com base no código exemplo fornecido. O servidor deve ser desenvolvido com base em Flask[1] e suportar o envio de ficheiros vídeo mp4 através de um *endpoint* na raiz do servidor.

Segundo passo é o servidor particionar o vídeo em *frames* e fazer o envio de *frames* individuais a um único *Worker*. O protocolo a usar nesta comunicação deve ser definido pelo grupo de trabalho e documentado num pequeno relatório. Neste protocolo são enviados frames e não caminhos para os ficheiros que contêm os frames.

Na etapa seguinte, o *Worker* deve realizar o trabalho útil pretendido (detectar objectos) com base no código exemplo fornecido.

Agora que temos um sistema funcional, deve generalizar para um sistema com múltiplos *workers*. O número de *workers* não deve estar definido em lado algum. Sempre que o processo *worker* é lançado, este deve-se registar junto do servidor e passar a fazer parte da *pool* deste.

Deve procurar optimizar o sistema ao máximo com o objectivo de processar o vídeo completo no menor tempo possível. No relatório deve incluir resultados de tempo de processamento para 1 *worker*, 2 *workers*, 4 *workers*.

Pará lá de informar o utilizador de alguma situação de contra-ordenação por presença de pessoas a mais no local, o servidor deve terminar o processamento do vídeo com algumas estatísticas:

- Número de frames processados
- Tempo médio de processamento por frame
- Número Total de pessoas detectadas em todos frames (somatório)
- Número Total de classes detectadas
- Top 3 de objectos detectados.

Tabela 1 - Example Server output

```
Frame 123: 13 <person> detected

Processed frames: 1300
Average processing time per frame: 100ms
Person objects detected: 12345
Total classes detected: 9
Top 3 objects detected: person, boat, car
```

Deve seguir o formato da Tabela 1 à risca para questões de avaliação.

Junto com este enunciado são fornecidas duas aplicações *python* (`video2image.py` e `object_detect.py`) que fazem uso de diversas bibliotecas de processamento de imagem como o *OpenCV* e o *TensorFlow*. Estes ficheiros devem de servir de exemplo, pode alterar e combinar este código como entender.

[1] https://flask.palletsprojects.com/en/1.1.x/

Os trabalhos que almejem notas superiores a 16 valores devem implementar funcionalidades redundantes, nomeadamente: se um *worker crashar* (ou for morto com um `*kill*`) o *frame* deve ser trabalhado por outro *worker*. Deve suportar mais do que um vídeo em concorrência.

Avaliação

A avaliação deste trabalho será feita através da submissão de código na plataforma GitHub classroom e de um relatório em formato PDF com não mais de 5 páginas colocado no mesmo repositório junto com o código e com o nome `relatorio.pdf`.

Está em avaliação o protocolo definido e documentado, assim como as *features* implementadas de acordo com os objectivos definidos na secção anterior.

Referências

[1] https://pjreddie.com/media/files/papers/YOLOv3.pdf

[2]https://github.com/YunYang1994/TensorFlow2.0-Examples/tree/master/4-Object_Detection/YOLOV3

[3] https://docs.python.org/3/library/tempfile.html