TP4
Ordonnancement de flux vidéo

Romain Raveaux

Objectif

Sensibiliser les étudiants à la diffusion par un réseau de vidéos à différents débits. Les serveurs multimédia sont amenés à diffuser des contenus à différents débits pour minimiser la bande passante ou s'adapter aux terminaux. Les vidéos à diffuser peuvent nécessiter différent temps de traitement par image en fonction du codec de décompression (mpeg2, mpeg4, …) ainsi que différents débits (frame rate).

Ordonnancement en ligne.

1°) Réaliser un programme capable de diffuser une vidéo à un nombre de frames par seconde donné.

Pour ce faire vous pourrez soit utiliser les thread python soit les timer python.
Voici un exemple d'utilisation des thread en python :

```
import numpy
import cv2
from threading import *
import time


class videoprocess(Thread):

    """Thread chargé simplement d'afficher une lettre dans la console."""

    def __init__(self, videoname,fps,id):
        Thread.__init__(self)
        self.videoname = videoname
        self.fps = fps
        self.id=id
        print('thread='+videoname+' fps='+str(fps))

    def run(self):
        """Code à exécuter pendant l'exécution du thread."""
        print('run thread='+self.id) #ne pas décommenter cette ligne

# Création des threads
```

```
thread_1 = videoprocess('videotestverylow.mp4',1,1)

# Lancement des threads
thread_1.start()
time.sleep(1) #warmup du thread
# Attend que les threads se terminent
thread_1.join()

cv2.destroyAllWindows()
```

Le nombre de frame par seconde sera choisi au lancement du thread.
Afficher le nombre de frame par seconde.
Quelle est la valeur maximum ?

2°) Réaliser un programme capable de diffuser plusieurs vidéos.

Chaque vidéo pourrait être jouée à un nombre de frame par seconde différent.
Chaque vidéo doit être jouée dans un thread qui lui est propre.
Afficher le nombre de frame par seconde sur chaque vidéo.
Scénario de test :
Imposer un nombre de frame par seconde égale à la valeur max trouvée précédemment.
Chaque thread jouera la vidéo 'videotestverylow.mp4'.
Créer et démarrer 5 threads, chaque démarrage de thread sera espacé de 15 secondes.
Que constatez vous ?
Votre système devra être capable de dire si le système (l'ensemble des vidéos) est « ordonnançable ». Calculer toutes les 5 secondes la formule suivante :

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

P_i = 1/fps période de l'événement en seconde
C_i = temps pour traiter une frame en seconde

3°) Implémenter l'ordonnancement RMS (Rate Monotonic Scheduling)

Rappeler les conditions pour pouvoir exécuter un ordonnancement RMS ?
Rappeler le principe de fonctionnement de RMS ?

Implémenter un ordonnanceur implique de pouvoir préempter un processus (un thread dans notre cas de simulation car en vrai il y a toujours l'ordonnanceur de l'OS).

Regarder le code ci-dessous et implémenter les méthodes permettant de mettre en pause un thread et de reprendre son exécution.

Créer un programme capable d'ordonnancer les threads selon le principe RMS.

Dans vos threads séparais la phase de traitement et la phase de sommeil.
Décomposer la phase de sommeil afin que cette dernière soit possiblement mise en pause par l'ordonnancement. La phase sommeil sera une série de petits 'sleeps'.
Les phases de traitements sont non-préemptables, insécable dans notre cas d'étude.
Quand le thread est en pause puis réveillé, vous devez être en mesure d'enlever du temps de sommeil le temps de pause.
Dans le programme principal vous pouvez suivre le pseudo code suivant :
Boucle infini :

Trouver le thread avec la plus grande priorité et n'étant pas entrain de dormir
Mettre en pause tous les threads sauf le thread prioritaire
Attendre que le thread prioritaire est fini son traitement et soit entré en phase de sommeil.
Tous les threads passe de « en pause » à « reprise ».

Tester votre algorithme avec 3 threads (fps1 =2,fps2=10,fps3=20).

```
class Monthread(threading.Thread):

  def __init__(self):
    threading.Thread.__init__(self)
    self.encore = True # =variable pour arrêter le thread sur demande
    self.enpause = False # variable pour mettre le thread en pause

    self.c = 0 # simple compteur pour affichage

  def run(self):

    while self.encore:

      while self.enpause:
        time.sleep(0.5)

      # activité du thread
      # ...
      self.c += 1
      print(self.c)
      # ...

  def pause(self):
    self.enpause = True

  def reprise(self):
    self.enpause = False

  def stop(self):
    self.encore = False

##################################################################
#################
```