UNIVERSIDAD DE VALLE
Infraestructuras Paralelas y Distribuidas – 750023C-01

PARCIAL I

Samuel Galindo Cuevas 2177491
Nicolás Herrera Marulanda 2182551
Christian David Vargas Gutiérrez 2179172

El código fuente del programa se encuentra alojado en el repositorio https://github.com/ChristianV2426/Parcial-I-Infraestructuras

REQUERIMEINTOS Y ESPECIFICACIONES

Para que la aplicación funcione correctamente, es necesario que la máquina en la que se ejecute tenga instalado:

- **Python** versión 3.8 o superior
- **yt-dlp** versión 2024.04.04 o superior
- **ffmpeg** versión 7.0 o superior

El archivo de entrada del programa debe ser un archivo de texto **.txt** que contenga 5 líneas, cada una con la URL de un canal de YouTube. Por ejemplo:

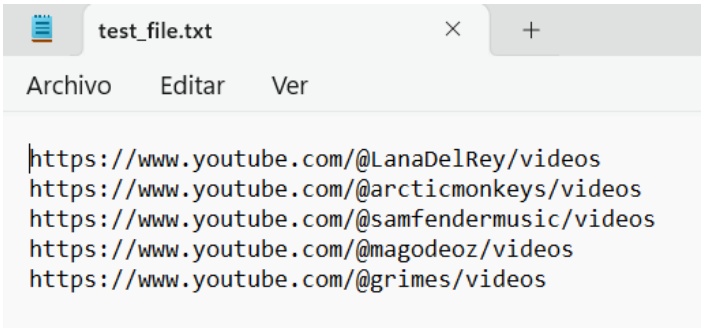

Imagen 1 – Ejemplo del archivo de entrada aceptado por el programa.

Los archivos de salida serán los audios de los 5 últimos vídeos de cada canal, guardados en la carpeta especificada en la ruta de descarga. Además, se creará un archivo **info.json** con la información de los vídeos descargados (título, fecha de subida a YouTube y fecha de descarga del vídeo). Por ejemplo:

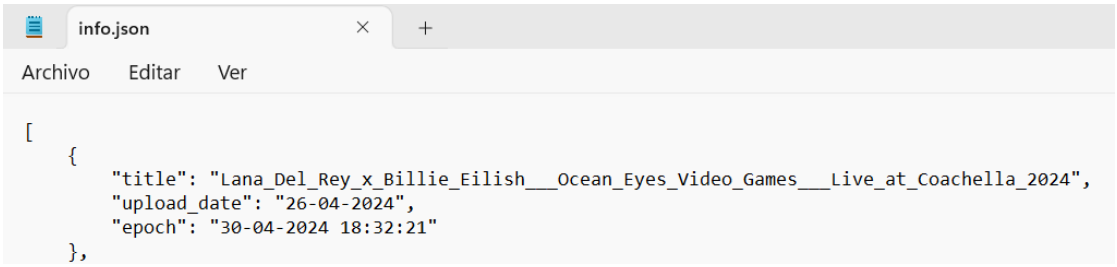

Imagen 2 – Ejemplo del archivo de salida info.json del programa

Para ejecutar el programa, se debe abrir una terminal desde la misma ubicación en la que se encuentra el archivo **main.py** y ejecutar el siguiente comando, con los argumentos y banderas en orden estricto:

python3 main.py <***ruta_archivo_entrada***> <***ruta_descarga***>
opcional: <***modo_ejecución***> <***número_hilos*** o ***número_procesos***>

Donde:

- <***ruta_archivo_entrada***> es la ruta del archivo .txt que contiene las URLs de los 5 canales de YouTube de los que se desean descargar los últimos 5 vídeos.

- <***ruta_descarga***> es la ruta en la que se guardarán los vídeos descargados. El programa creará una carpeta a este nivel con el nombre de */audio_extraction_at_<fecha_descarga>*.

- <***modo_ejecucion***> es el modo en el que se desea ejecutar el programa. Puede ser "**-t**" (**multithreading**) o "**-p**" (**multiprocessing**). Si no se especifica, se ejecutará en modo secuencial.

- <***número_hilos*** o ***número_procesos***> si se especifica el <*modo_ejecucion*>, se debe indicar el número de hilos o procesos a utilizar.

FUNCIONAMIENTO DEL PROGRAMA

El script que define las tareas que realiza cada trabajador (hilo o proceso) se encuentra en la función script dentro del paquete modules.script.

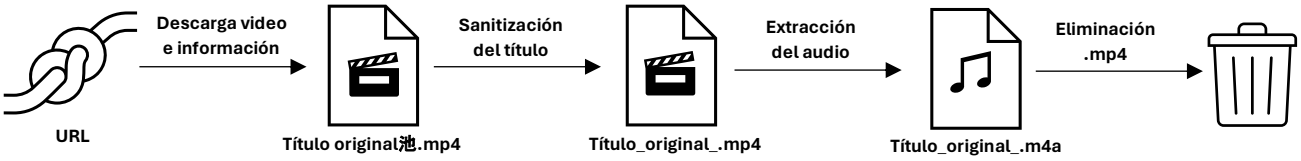

Imagen 3 – Representación gráfica de las tareas que lleva a cabo cada trabajador en la ejecución paralela

1. En este script lo primero que se hace es utilizar yt-dlp para descargar el video pasado como argumento, en formato mp4. Adicionalmente, en una lista compartida por todos los hilos o procesos se guara un objeto json de Python toda la información relacionada al vídeo (título, fecha de subida a Youtube y fecha de descarga local).

2. A continuación se realiza una sanitización del título del archivo mp4, para evitar espacios y caracteres que después le impidan al sistema operativo encontrar el archivo.

3. El siguiente paso involucra la extracción del audio desde el archivo de vídeo utilizando ffmpeg. El audio se guarda en un archivo con el mismo título pero con extensión m4a.

4. Como última etapa, se elimina el video .mp4 del fichero de descarga.

Una vez todos los trabajadores han terminado de ejecutar el script para los vídeos que se le asignan, entonces el hilo principal (main) traslada la información guardada en la lista de json (paso 1) a un archivo local info.jso. Finalmente, se imprime en consola un mensaje anunciando la terminación del programa y las métricas de tiempo obtenidas.

La asignación de responsabilidades a cada trabajador se programó utilizando ciclos for. Por ejemplo, para la ejecución paralela con 4 hilos, el hilo 1 está encargado de ejecutar el script para los videos en la posición 0, 4, 8, 12, 16, 20 y 24 de la lista de videos a descargar (imagen 4).

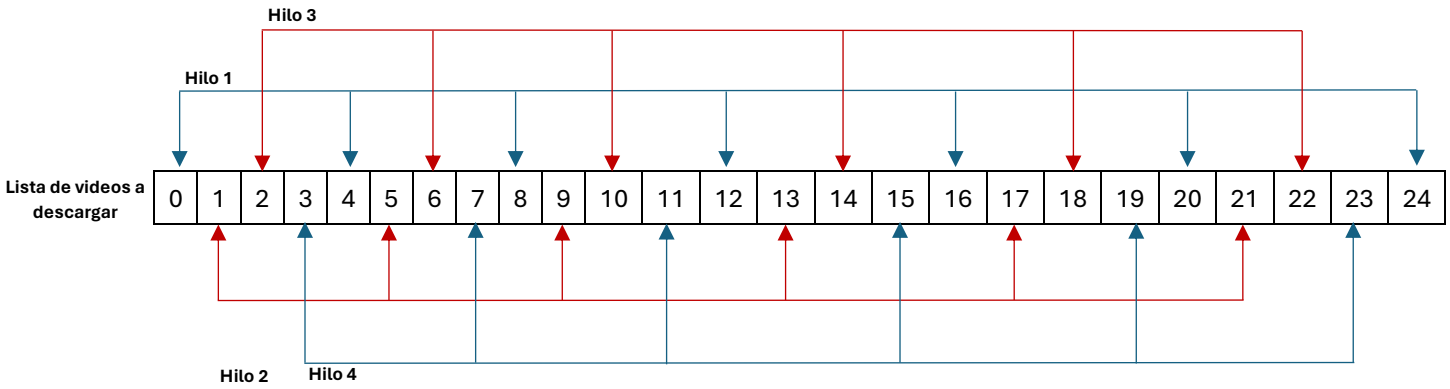

Imagen 4 – Distribución de responsabilidades a cada trabajador en una ejecución paralela con 4 hilos.

TIEMPOS DE EJECUCIÓN

El computador utilizado para las pruebas tiene las siguientes características: procesador Intel Core i5-1135G7 @ 2.40GHz 8 Cores, 12 GB de RAM. La red de internet utilizada tiene una velocidad de descarga 353.9 Mbps y de 400.7 Mbps de subida (se empleó https://www.speedtest.net/ para medirla). El programa se ejecutó utilizando los canales de YouTube mostrados en la imagen 1 (los videos descargados tenían en promedio una duración entre los 3 y 6 minutos).

Ejecución secuencial		
Prueba	**Tiempo (s)**	**Tiempo (min)**
1	183.26	3.05
2	183.92	3.07
3	186.32	3.11
4	195.67	3.26
Promedio	**187.30**	**3.12**

Tabla 1 – Tiempos de ejecución en la versión secuencial

Ejecución multihilos			
Número de hilos	**Prueba**	**Tiempo (s)**	**Tiempo (min)**
4	**1**	56.85	0.95
	2	56.39	0.94
	3	56.77	0.95
	4	53.70	0.90
	Promedio	**55.93**	**0.93**
8	**1**	42.76	0.71
	2	41.22	0.69
	3	43.57	0.73
	4	41.59	0.69
	Promedio	**42.29**	**0.70**
16	**1**	32.35	0.54
	2	35.29	0.59
	3	35.47	0.59
	4	34.06	0.57
	Promedio	**34.29**	**0.57**

Tabla 2 – Tiempos de ejecución en la versión multihilo

Ejecución multiproceso			
Número de procesos	**Prueba**	**Tiempo (s)**	**Tiempo (min)**
4	**1**	55.69	0.93
	2	56.36	0.94
	3	56.35	0.94
	4	59.80	1.00
	Promedio	**57.05**	**0.95**
8	**1**	40.40	0.67
	2	43.32	0.72
	3	42.28	0.70
	4	41.45	0.69
	Promedio	**41.86**	**0.70**
16	**1**	36.02	0.60
	2	36.87	0.61
	3	37.46	0.62
	4	36.11	0.60
	Promedio	**36.62**	**0.61**

Tabla 3 – Tiempos de ejecución en la versión multiproceso

ANÁLISIS DEL PARALELISMO LOGRADO

En la siguiente tabla se resumen las ganancias en tiempo de ejecución logradas al paralelizar utilizando hilos y procesos. Se recuerda que el tiempo promedio secuencial obtenido fue de 187.30 segundos (tabla 1).

Tipo de ejecución	n	Tiempo prom (s)	Speedup	Eficiencia (%)	f (%)
Multihilo	4	55.93	3.35	83.7	93.5
	8	42.29	4.43	55.4	88.5
	16	34.29	5.46	34.1	87.1
Multiproceso	4	57.05	3.28	82.1	92.7
	8	41.86	4.47	55.9	88.7
	16	36.62	5.11	32.0	85.8

Tabla 4 – métricas del paralelismo logrado con las ejecuciones multihilo y multiproceso

Para el cálculo del speedup se utilizó el cociente entre el tiempo secuencial sobre el tiempo paralelo.

$$\text{Speedup} = \frac{\text{t}_{\text{secuencial}}}{\text{t}_{\text{paralelo}}}$$

La eficiencia resulta del cociente entre el speedup y el número de trabajadores (hilos o procesos) utilizados en la versión paralela.

$$\text{Eficiencia} = \frac{t_{secuencial}}{t_{paralelo} * n_{trabj}}$$

Finalmente, para hallar el porcentaje de la aplicación paralelizable f, se empieza con la ley de Amdahl y se resuelve f.

$$\text{Speedup} = \frac{1}{1 - f + \frac{f}{n_{trabj}}}$$

$$f = \frac{\frac{1}{s} - 1}{\frac{1}{n_{trabj}} - 1}$$

Con respecto al performance de las versiones paralelas (tabla 4), como es de esperarse, se obtiene un aumento en la velocidad de ejecución con respecto a la versión secuencial (speedup mayor a 1). Para la ejecución multihilo, el speedup máximo se alcanza con 16 hilos, con un valor de 5.46, lo que indica que la ejecución es aproximadamente 5.5 veces más rápida que la ejecución secuencial en esa configuración. Similarmente, para la ejecución multiproceso, el speedup máximo también se alcanza con 16 trabajadores (procesos), con un valor de 5.11.

Sin embargo, el speedup no es la única métrica que mide la calidad de la paralelización. También hay que analizar la eficiencia, que representa qué tan bien se están utilizando los recursos de procesamiento adicionales. Tanto en el caso de la ejecución multihilo como en la multiproceso, la eficiencia disminuye a medida que aumenta el número de hilos o procesos.

Esto indica que a pesar de que sí hay una ganancia en tiempo de ejecución al aumentar el número de trabajadores, probablemente cada trabajador esté siendo subutilizado y contribuyendo menos al rendimiento general del programa. Esto podría deberse al hecho de que los videos descargados no son lo suficientemente largos como para justificar un aumento en el número de líneas que trabajan paralelamente. De hecho, las ejecuciones con 16 hilos / procesos tienen apenas una eficiencia cercana al 30%, lo que es muy pequeño a comparación con la eficiencia de las ejecuciones con 4 hilos / procesos que

superan al 80%. Podemos concluir que para este volumen de trabajo utilizar más de 4 trabajadores implicaría una inversión de recursos que no justifica la ganancia en reducción de tiempos de ejecución.

Por último, la eficiencia relativa, f, muestra el porcentaje de aplicación que en teoría es paralelizable. A pesar de que al aumentar el número de trabajadores, f disminuye, es positivo notar que al comparar los resultados para el mismo número de trabajadores en las ejecuciones multihilo y multiproceso hay consistencia. Los resultados globales sugieren que alrededor del 89% del programa es paralelizable, lo que explica los buenos rendimientos que se obtuvieron con ambas versiones paralelas.