UNIVERSIDAD
DE SANTIAGO
DE CHILE

Departamento de Matemática y Ciencia de la Computación

Trabajo Instancias EC2 en AWS

Ingeniería de software 3

Segundo Semestre 2022

Ingeneria en software 3

Jeremy Betancourt Méndez
Alexis Bolados del Castillo
Pedro Figueroa Cerda
Sergio Meirone Aburto
Ignacio Sanhueza Zuñiga

Licenciatura en Ciencia de la Computación

1 Introducción

En el presente documento se describe el desarrollo del trabajo final de *Ingeniería de software 3*, cuyo objetivo consiste en realizar una prueba de rendimiento a un clúster de instancias con la API de Amazon Web Services (AWS). Para tal propósito se debe escoger, en primer lugar, entre dos librerías, **ffmpeg** y **mencoder**, para la implementación de una serie de seis operaciones de transcodificación, las cuales son:

1. Cambio de resolución: A full HD, HD y un tamaño menor, a elección (800x600, 640x480, etc).

2. Cambio de tasa de aspecto 16:9 a 5:4 y viceversa.

3. Extracción de un fragmento, escogiendo instantes de inicio y final.

4. Recorte de un área rectangular del video, escogiendo punto superior izquierdo e inferior derecho.

5. Extracción de audio de parte del video a formato raw.

6. Codificación del audio a flac/ogg/mp3.

En segundo lugar, se requiere correr un algoritmo distribuido en el mismo clúster de instancias EC2 con la finalidad de corroborar el rendimiento de una prueba distribuida. Para ello se crea una primera instancia de la capa gratuita que funciona como base comparativa, una segunda instancia con mayores recursos, y en tercer lugar un clúster de instancias, para la implementación paralela. El algoritmo seleccionado es la suma de n números aleatorios los cuales se resuelven de manera distribuida entre los nodos definidos.

Nuestra elección sobre la herramienta a utilizar en el primer ejercicio es la librería **ffmpeg** por las razones de que tal librería es más simple de ser usada ya que su instalación es más fácil en comparación a **mencoder**, a pesar de que este utiliza menos memoria para su instalación respecto a **ffmpeg**. Además, en cuanto a eficiencia de los algoritmos que utiliza están bastante a la par, o no llegaríamos a experimentar diferencia alguna, como por ejemplo, **mencoder** puede trabajar de manera más eficiente con 3 búffer que con 2, pero su configuración es muy engorrosa y para los propósitos de generar un estrés de rendimiento a la máquina virtual no vamos a preferir lo más eficiente sino lo más extenuante.

2 Arquitecturas Seleccionadas

Para la arquitectura seleccionada, el equipo de trabajo ha manipulado y utilizado **ec2**, específicamente la instancia disponible para la capa gratuita **t2.micro**. Las instancias T2 de ec2 proveen de rendimiento de nivel base para la CPU. Las instancias t2 disponen de una CPU virtual Intel Xeon Scalable de hasta 3,3 GHz y un Gigabit de memoria RAM.

Para la ejecución de la tarea de transcodficación, se lanzó una sola instancia t2.micro, con el sistema operativo Ubuntu. La razón para lanzar una sóla instancia es que no fue posible generar la conexión entre diferentes instancias, lo más lejos que se llegó fue configurar las instancias para poder realizar un ping entre ellas asegurando que exista la conexión, el detalle sobre el registro de la velocidad de ping entre una instancia y la otra está en el apéndice 1.

3 Discusión de la elección de arquitectura

De la vasta cantidad de instancias que ofrece ec2, solo hay dos tipos que están disponibles para la capa gratuita: la **t1.micro** y la **t2.micro**. Nuestra elección de la instancia **t2.micro** está fundamentada en que las instancia **t1.micro** tienen prestaciones similares, excepto por la memoria RAM, ya que solo cuenta con 0.612 Gb, por lo que consideramos que la memoria extra de **t2.micro** es más apropiada para la prueba de transcodificación.

Aún así, la arquitectura seleccionada fue en un comienzo la arquitectura **t1.micro** para poder efectuar una comparación entre la eficiencia de ésta con la arquitectura **t2.micro**. Pero no fue posible montar la máquina sin producir errores en la configuración ya que no era ápto para ser usado con el sistema operativo **Amazon Linux**. Se pretendía el uso de tal SO para el efecto de la configuración automática del cluster ya que es la única que permite lanzar un script inicial al iniciar la isntancia. Sin embargo, tal pretención no prosperó dado a la cantidad enorme de errores que incluso con la arquitectura seleccionada **t2.micro** no fue posible de realizar. Finalmente nos decantamos por el uso del sistema operativo de **Ubuntu** por la simpleza en su uso y configuración manteniendo como arquitectura de las instancias la **t2.micro** de la capa gratuita.

Por otro lado, respecto al sistema operativo, hicimos algunas pruebas con amazon linux, ubuntu y debian. La elección de ubuntu por sobre amazon linux está justificada por el hecho de que no pudimos instalar la herramienta "exiftool" a través del gestor de paquetes "yum" de amazon linux. Puesto que esta herramienta es necesaria para realizar la prueba de transcodficación, preferimos utilizar ubuntu y "apt" que sí permite instalar todas las dependencias necesarias para la prueba.

4 Configuración de AWS e instancias EC2

Se utilizan 3 instancias alojadas en el norte de Virginia con las características en la selección de arquitectura.

Estas constan de IP públicas, las cuales permiten un manejo en un entorno mas familiar, ya que se pueden llamar desde la consola de sistemas basados en Unix (específicamente en nuestro caso, en Ubuntu y Debian), donde mediante el protocolo ssh y una clave en RSA con extensión ".pem", hacen posible esta forma de trabajar.

5 Configuración de clúster

La configuración del clúster se realiza mediante la instalación de Jenkins, que es un servidor de automatización open source, basado en Java. Entonces, mediante esta herramienta se construye un sistema distribuido que consta de dos nodos esclavos y un nodo máster. Para la instalación, es importante mencionar que se requiere haber instalado Java en las instancias EC2, para luego realizar la instalación de Jenkins, posteriormente se configuran los firewall UFW de cada nodo, y se crea una cuenta en la dirección `http://3.88.230.202:8080`.

La configuración no estuvo exenta de problemas. Se tuvo que pasar por varias iteraciones en las cuales el punto más crítico fue el tener que abrir los puestos TCP 8080 para la conexión por medio de éste a la IP pública. También se tuvo que registrar las IP's privadas para correr el código distribuído en base a un archivo de hosts sin éxito, por lo cual se intentó también con las IP's públicas y variados formatos del archivo, sin éxito.

6 Costo asociados y Desempeño según carga en uso de procesadores, memoria RAM y almacenamiento

Cada nodo implementado es configurado con elementos de la capa gratuita, ya sea en su arquitectura y características de comunicación (públicas). Para configurar el clúster, utilizando el servidor Jenkins, se intenta levantar un grupo de seguridad con 1 máster y 2 esclavos con propiedades idénticas para las 3 instancias, pero el resultado de la implementación no es satisfactorio, puesto que se presentan dificultades a la hora de ingresar claves y las direcciones públicas de los nodos, y estos mismos, con fallas en la conexión dentro de AWS.

Si bien no se han efectuado cobros extra al proyecto, tal limitación tuvo un impacto directo en la satisfacción de los requerimientos. La forma correcta de establecer la configuración del cluster entre las instancias EC2 es por medio de un grupo de ubicación. Sin embargo, no existen arquitecturas pertenecientes a la capa gratuita que permitan ser añadidas a un grupo de ubicación. Tal limitación nos llevó a intentar configurar sin éxito de manera manual las instancias.

7 Detalle de las pruebas efectuadas

En la presente sección, se entregan los algoritmos realizados:

Script de Prueba para el Rendimiento en bash.

para efectuar esta comprobación se utilizaron dos paquetes:

- [**exiftool**] - que fue utilizado para poder obtener la metadata del vídeo, y así identificar tanto resolución como relación de aspecto, y la duración del mismo, dentro de variables para ser utilizadas en el script

- [**ffmpeg**] - es un codex multimedia, su función es convertir audio y video a los diferentes tipos de formatos que existen hoy, tanto como los formatos de audio y video, otra funcionalidad que tiene es que puede editar y transmitir por streaming los archivos multimedia

Comandos utilizados desde **ffmpeg** para la implementación:

```
ffmpeg -i <input.mp4$> -vf scale=<value>:<value> -preset slow -crf 18 <output>

ffmpeg -i <input.mp4> -aspect 4:3 <output>

ffmpeg -i <input.mp4> -aspect 16:9 <output>

ffmpeg -i <input.mp4> -map 0 -default\_mode
infer\_no\_subs -ss <start> -to <end> -c copy <output>

\item   ffmpeg -i <input.mp4> -vf crop='pixel1':'pixel2' <output>

\item   ffmpeg -i <nput.mp4> -vn -f f32le -acodec
pcm\_f32le -ac 2 output-audio.raw

\item   ffmpeg -f f32le -ac 2 -i output-audio.raw $<$output.'format'>
```

- Para la implementación del programa utilizando memoria privada, la construcción fue basada en la utilización de la librería mpich, previamente instalada en las instancias, con el fin de poder compilar y ejecutar el algoritmo "distsum.c", donde se destaca la utilización de los protocolos de envío y recepción de arreglos de números.

- Los protocolos mencionados, dentro de la librería toman el nombre de MPI_Send y MPI_Recv, donde ambos casos requieren el arreglo o la dirección de este, el tamaño, el tipo de variable, la recepción o el envío (o viceversa) de la recolección de datos.

- El Algoritmo se envía por parámetro, con todo el preámbulo de ejecución de la distribución ("mpirun" y la cantidad o descripción de nodos/host), el ejecutable que se compila, la cantidad de sumas que realiza y el modo de ejecución, Silencioso o verboso (-S y -V). Dentro del algoritmo los números a sumar son ingresados de manera pseudoaleatoria utilizando los procesos y el tiempo de ejecución de los nodos.

Adicionalmente se han verificado las entradas para evitar los errores de ejecución con las herramientas seleccionadas.

8 Conclusiones

En el trabajo se logran implementar algoritmos en bash que utilizan la transcodificación de "ffmpeg", para el tratamiento de vídeo, donde podemos obtener un cambio de resolución, también en la tasa de aspecto, una extracción de fragmento y recorte de área, también la extracción de audio y conversión a formato "mp3", "ogg" o "FLAC".

También se implementa en "C" el algoritmo de la suma, utilizando la librería de "mpi", donde este recibe parámetros de entrada, la cantidad de sumas y el modo de ejecución, para la verificación de resultados y tiempo de ejecución.

La dificultad del trabajo, fue al utilizar el servicio de AWS, puesto que no siempre cada instancia permanecía conectada, y no en todas las opciones de S.O. se podían utilizar la consola virtual, o mediante el protocolo SSH tampoco se lograba una conexión estable en estos. Otra de las dificultades sin solución, fue la implementación de los nodos como un clúster, puesto que el servidor jenkins no reconocía las credenciales necesarias de las instancias de AWS. Incluso sin la utilización de herramientas externas como jenkins, la configuración de un cluster es un desafío por si mismo merecedor de su propio trabajo aparte.

9 Apéndice 1: log de velocidad en ping entre instancias AWS

Se puede evidenciar en el archivo adjunto **tiempos_ping.txt** que el tiempo promedio de la conexión entre las instancias es de 0,6 segundos para el ping.