Mitsiu Alejandro Carreño Sarabia - E23S-18014

ANÁLISIS TOPOLÓGICO DE DATOS - Benchmark Dask vs Pandas-Chunks

Objetivo

El siguiente código permite realizar una comparación de performance entre dos tecnologías de manejo de big data, una empleando dask con concurrencia e hilos y la otra usando pandas chunks en un solo hilo. Durante la medición de performance ambas tecnologías ingresan un renglón a la vez en una base de datos postgres. Finalmente cuando se han guardado todos los renglones en la base de datos se mide el tiempo que tomó realizar el procesamiento. En la sección de resultados se exponen los resultados obtenidos.

Estructura

El proyecto cuenta con 4 carpetas:

- chunks - Contiene el código necesario para a) Generar el contenedor para procesar en chunks de pandas
  b) Generar el contenedor de postgres que almacenará la información
  c) Código para realizar el procesamiento del csv en chunks de pandas y medir el tiempo de ejecución
- dask - Similar a la carpeta `chunks` contiene el código para: a) Generar el contenedor para procesar en dask
  b) Generar el contenedor de postgres que almacenará la información (usa el mismo puerto que el contenedor de postgres en chunks)
  c) Código para realizar el procesamiento del csv en dask y medir el tiempo de ejecución
- data - Contiene el código para generar 5 archivos csv con longitudes variables cada uno de estos archivos puede ser usado para el benchmark, los archivos son: a) full_benchmarking_data.csv - 307,200,001 renglones
  b) 10k_benchmarking_data.csv - 10,000 renglones
  c) 100k_benchmarking_data.csv - 100,000 renglones
  d) 1M_benchmarking_data.csv - 1,000,000 renglones
  e) 50M_benchmarking_data.csv - 50,000,000 renglones
- shared_scripts - Contiene código que maneja la conexión a la base de datos y es compartido entre el contenedor de dask y el contenedor de chunks, además contiene las credenciales para configurar y acceder a la base de datos, en un ambiente productivo usualmente estas credenciales estarán alojadas en algún servicio como infisical o aws secret manager, pero lo importante es que estén desacopladas del código.

Ejecutar proyecto

1. Generar grandes cantidades de información para medir performance
   Dentro del directorio `data` se deberá ejecutar el archivo `gen_data.py` con el siguiente comando:

```
cd data
podman run --rm -v .:/usr/src/app:Z -w /usr/src/app \
python:3.9.19-alpine3.20 sh -c "pip install pandas && python gen_data.py"
```

El script correrá alimentando el archivo `full_benchmarking_data.csv` y una vez completado generará los archivos más pequeños (`10k_benchmarking_data.csv`, `100k_benchmarking_data.csv`, `1M_benchmarking_data.csv`, `50M_benchmarking_data.csv`) todos estos archivos se generarán en la carpeta `data`.

2. Elegir el archivo de la carpeta `data` que se desea someter a prueba (por default está `10k_benchmarking_data.csv`) y reemplazar en "FILE_PATH" en los archivos `chunks/container-compose.yml` y `dask/container-compose.yml`

3. Elegir la carpeta `chunks` o `dask` y ejecutar el comando:

```
cd chunks
podman-compose up
```

El programa iniciará ejecución y al finalizar mostrará un mensaje "Completed execution in X seconds".
Una vez finalizado, se deberá correr el comando

```
podman-compose down
```

Entonces podremos cambiar de carpeta y ejecutar en la otra técnología

```
cd ../dask
podman-compose up
```

Resultados

Se realizaron dos experimentos en el primero se comparó dask y chunk insertando en una base de datos de postgres un renglon a la vez

Insertando en postgres	Chunks	Dask
10k_benchmarking_data.csv	10.1904	11.2390
100k_benchmarking_data.csv	116.2252	115.7151
1M_benchmarking_data.csv	1220.0703	1040.1997

En el segundo se comparó dask y chunk contabilizando renglones (sin insertar en la base de datos)

Conteo de renglones	Chunks	Dask
10k_benchmarking_data.csv	0.0957	0.0330
100k_benchmarking_data.csv	0.9444	0.1891
1M_benchmarking_data.csv	9.3573	1.6793
50M_benchmarking_data.csv	513.6735	106.4634
full_benchmarking_data.csv	3197.2531	646.4224

Ambas tablas con métricas en segundos.

Conclusiones

Se puede apreciar que conforme aumenta el tamaño del archivo a procesar Dask comienza a ofrecer mejor rendimiento, pero no por tanto margen, la razón de ello es que independientemente de la técnica de procesamiento el insertar en la base de datos es un cuello de botella. Dask ofrece mejor rendimiento porque al emplear más de un hilo para procesar, termina realizando más de una conexión a la base de datos.
Para la versión que no insertaba en la base de datos los tiempos generales bajaron drasticamente, validando que la base de datos es un cuello de botella, solo en recorrer el archivo Dask fue consistentemente más rápido gracias a la paralelización que ofrece, además de manejar bien una variable global compartida.
Cabe destacar que aun queda amplio margen de mejora en el análisis, ya que ambos perfiles, dask y chunks empleaban muy poco cpu (~10%) y como lo comenté previamente el principal problema es las operaciones de lectura/escritura de RAM a disco. Además creo que es posible empujar un poco más la manera en que Dask procesa para obtener mejores prestaciones.
Finalmente como posible área de mejora está en adaptar el código para permitir ingresar datos en lote en lugar de uno a la vez, eso se espera que mejore el tiempo de ejecución de ambas tecnologías.

Links y referencias

- https://stackoverflow.com/questions/54148429/how-to-read-a-csv-and-process-rows-using-dask
- https://stackoverflow.com/questions/56188573/permission-issue-with-postgresql-in-docker-container
- https://docs.dask.org/en/stable/deploying-docker.html