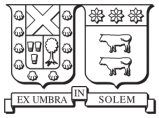

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Trabajo Práctico 2

INF356 - 2025-1 - 200

Computación Distribuida para Big Data

15 de mayo de 2025 - v1.0

Wedge Antilles | 123456789-0
wedge.antilles@rebelalliance.com

Índice

Índice de tablas

Índice de figuras

Índice de fragmentos de codigo

Instrucciones

El presente documento corresponde a la plantilla para presentar las informaciones que deben ser proveídas para evaluar la entrega.

Todos los textos en rojo a lo largo de la plantilla, junto con esta página de instrucciones, deben ser eliminadas antes de la compilación final.

1. Map Reduce

Una vez terminada esta sección puede destruir su cluster de Hadoop

1.1. Implementación extendida de wordcounter

Utilizando el ejemplo WordCounter de Hadoop, implemente un programa que reciba un archivo y cuente las ocurrencias de cada palabra en ese texto. Debe considerar lo siguiente:

- Se considera palabra cualquier secuencia de carácteres separada por espacios
- Una secuencia de carácteres especiales no corresponde a una palabra debe ser descartada
- Se debe eliminar cualquier puntuación al inicio o final de la palabra, pero se debe conservar puntuaciones que estén entre carácteres
- Las palabras se deben contar por su equivalente en letras minúsculas

```
public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable> {
  public void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {
    // Code
  }
}
```

Código 1: Código utilizado en la implementación del mapper el wordcounter extendido

```
public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable> {
  public void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {
    // Code
  }
}
```

Código 2: Código utilizado en la implementación del reducer el wordcounter extendido

1.2. Implementación de selector de columna

Utilizando el ejemplo WordCounter de Hadoop, implemente un map-reduce que acepte como parámetro un único número entero denominado `select_column`. El map-reduce debe procesar un archivo con el formato **sw-script-e04.txt** y generar archivo que sólo contenga la columna indicada por el parámetro de entrada. Las columnas se cuentan desde 0. El número de columnas corresponde al registro que tenga la mayor cantidad de columnas. Si un registro no tiene valor para la columna indicada, se retorna un string vacío. Si el usuario indica cualquier número de columna fuera de los índices posibles, el archivo de salida debe ser igual al archivo original.

Note que el archivo **sw-script-e04.txt** es un archivo se secuencias de strings separadas por espacios, donde el inicio y final de cada string comienza y termina con comillas dobles. Con lo anterior, las comillas del inicio y final de cada columna corresponden a marcadores para leer el contenido, y por lo tanto no son parte del contenido.

```
public static class TokenizerMapper
    extends Mapper<Object, Text, Text, IntWritable> {
  public void map(Object key, Text value, Context context)
      throws IOException, InterruptedException {
    // Code
  }
}
```

Código 3: Código utilizado en la implementación del mapper del selector de columna

```
public static class IntSumReducer
    extends Reducer<Text,IntWritable,Text,IntWritable> {
  public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
    // Code
  }
}
```

Código 4: Código utilizado en la implementación del reducer del selector de columna

1.3. Tiempo de ejecución

Utilizando el filtro desarrollado en el punto anterior, genere un archivo que sólo contenga el contenido de la columna 2 (los diálogos). Sobre este archivo, analice el tiempo que demora el proceso wordcount extendido. Realice el análisis con las primeras 1, 10, 100, 500 y 1000 entradas del archivo. Realice el experimento 5 veces para cada cantidad de entradas de modo de obtener una idea sobre la dispersión del tiempo de ejecución.

	Promedio[s]			Desviación[s]		
Entradas	Real	User	Sys	Real	User	Sys
1						
10						
100						
500						
1000						

Tabla 1: Tiempos de ejecución de WordCount extendido

Sobre estos resultados, discuta cual es el tiempo de overhead debido a la utilización de map-reduce para realizar los cálculos.

2. Spark

2.1. Uso de training-bigdata-002

Demuestre que ha logrado crear un cluster de spark, testear el cluster y destruir el cluster de acuerdo a las instrucciones entregadas en `https://github.com/ptoledo-teaching/training-bigdata-002`

2.2. Escalamiento vertical y horizontal

Modificando el archivo de configuración de deployment del cluster de Spark, debe desplegar 5 configuraciones para medir el impacto que tienen el escalamiento vertical y horizonal en el tiempo de procesamiento. A cada estudiante le corresponderá una serie de configuraciones diferentes que se puede obtener de la siguiente lista:

- **19500967-8**: 2 x t3.large, 4 x t3.small, 8 x t3.large, 8 x t3.large, 8 x t3.small, 4 x t3.medium, 6 x t3.micro, 8 x t3.large, 2 x t3.micro, 4 x t3.medium
- **20068377-3**: 6 x t3.micro, 6 x t3.micro, 2 x t3.micro, 6 x t3.large, 10 x t3.small, 2 x t3.micro, 2 x t3.large, 2 x t3.medium, 10 x t3.large, 2 x t3.micro
- **20241011-1**: 4 x t3.micro, 10 x t3.micro, 8 x t3.micro, 4 x t3.large, 8 x t3.medium, 10 x t3.medium, 6 x t3.medium, 4 x t3.large, 4 x t3.medium, 6 x t3.medium
- **20298815-6**: 2 x t3.small, 10 x t3.medium, 10 x t3.large, 8 x t3.medium, 6 x t3.small, 6 x t3.medium, 2 x t3.micro, 2 x t3.small, 6 x t3.small, 2 x t3.large

- **20430363-0**: 4 x t3.micro, 4 x t3.small, 10 x t3.medium, 6 x t3.large, 8 x t3.medium, 10 x t3.micro, 4 x t3.small, 8 x t3.small, 10 x t3.large, 8 x t3.medium
- **20632334-5**: 6 x t3.medium, 4 x t3.medium, 4 x t3.medium, 6 x t3.large, 8 x t3.micro, 6 x t3.small, 10 x t3.large, 10 x t3.large, 4 x t3.small, 2 x t3.micro
- **20762701-1**: 8 x t3.micro, 2 x t3.medium, 6 x t3.micro, 10 x t3.small, 10 x t3.small, 10 x t3.medium, 4 x t3.large, 4 x t3.medium, 2 x t3.small, 2 x t3.large
- **20781979-4**: 10 x t3.large, 6 x t3.medium, 6 x t3.large, 2 x t3.small, 10 x t3.small, 2 x t3.medium, 10 x t3.small, 4 x t3.small, 8 x t3.medium, 8 x t3.large
- **20886083-6**: 2 x t3.large, 6 x t3.large, 4 x t3.micro, 8 x t3.micro, 4 x t3.large, 6 x t3.micro, 4 x t3.micro, 6 x t3.large, 6 x t3.large, 4 x t3.micro
- **20920259-K**: 4 x t3.small, 8 x t3.micro, 2 x t3.large, 8 x t3.medium, 6 x t3.micro, 8 x t3.small, 8 x t3.large, 2 x t3.small, 10 x t3.large, 4 x t3.micro
- **20966993-5**: 8 x t3.small, 6 x t3.small, 2 x t3.small, 10 x t3.large, 4 x t3.medium, 4 x t3.micro, 6 x t3.medium, 8 x t3.medium, 6 x t3.small, 8 x t3.micro
- **20969314-3**: 4 x t3.large, 10 x t3.small, 8 x t3.micro, 4 x t3.large, 4 x t3.medium, 10 x t3.micro, 10 x t3.micro, 8 x t3.small, 8 x t3.large, 4 x t3.large
- **21095788-K**: 4 x t3.micro, 4 x t3.small, 10 x t3.medium, 2 x t3.medium, 2 x t3.micro, 6 x t3.medium, 6 x t3.large, 10 x t3.small, 8 x t3.medium, 6 x t3.large
- **21127094-2**: 4 x t3.large, 6 x t3.small, 2 x t3.large, 2 x t3.small, 8 x t3.small, 2 x t3.medium, 10 x t3.medium, 2 x t3.small, 8 x t3.small, 10 x t3.medium
- **26799666-0**: 2 x t3.medium, 10 x t3.micro, 6 x t3.small, 2 x t3.medium, 10 x t3.micro, 2 x t3.medium, 6 x t3.small, 2 x t3.micro, 2 x t3.medium, 8 x t3.large

La cantidad de máquinas se refiere a la cantidad de máquinas trabajadoras. Usted debe desplegar el cluster para cada una de las configuraciones que le corresponden, ejecutar el test-000 (esto es necesario ya que este paso instala ciertas librerías que tienen un impacto relevante en el tiempo de ejecución) y luego medir el tiempo que demora el test-001. El tiempo debe ser reportado en `https://forms.gle/gGVc2wkqc6FzfyU99`. Debe completar el formulario para cada una de las 5 configuraciones solicitadas. En los casos de quienes tienen 2 veces asignada la misma configuración, se debe medir 2 veces y reportar 2 veces el tiempo requerido para ejecución.

Con la información obtenida se debe completar la tabla 2. En la columna de **costo** debe calcular el costo que tuvo la ejecución en USD tomando como referencia los precios on-demand por hora disponibles en `https://aws.amazon.com/ec2/instance-types/t3/`.

Maquinas	Tipo	Tiempo[s]	Costo[USD]

Tabla 2: Tiempos de ejecución para medición de impacto de escalamiento

Los datos recopilados entre todos los estudiantes estarán disponibles en `https://docs.google.com/spreadsheets/d/1t53S2s0knpQnZl9EcZr2ZHWOg_qTCcNp6Kjpg7g1Wlo/edit?usp=sharing`. En base a esta información usted debe desarrollar una figura que permita apreciar el efecto multidimensional del escalamiento vertical y horizontal en el tiempo de ejecución del proceso. Debe considerar a lo menos 10 pares diferentes maquina/cantidad para poder desarrollar esta figura (deberá esperar a que exista esta cantidad mínima de información para poder proceder).

Figura 1: Efecto del escalamiento vertical y horizontal en el tiempo de ejecución de test-001.sh Tiene libertad para elegir la metodología de visualización que le parezca más apropiada

Discuta sobre como afecta el escalamiento horizontal y vertical el tiempo de proceso del test-001, considerando tanto a la información que recopiló con sus experimentos, la información obtenida desde los experimentos de los otros participantes de la asignatura, y el impacto de la dispersión de las mediciones.

3. Procesamiento de datos

3.1. Extracción, transformación y carga

Desarrolle un programa basado en el test-001 que permita la extracción, transformación y carga (extraction, transformation and load, normalmente denominado ETL) del dataset de la asignatura. El set total de datos de la asignatura corresponde al archivo de 7.2[GB] disponible en `s3://utfsm-inf356-datasets/vlt_observations/vlt_observations.csv`. Este dataset ha sido dividido en 20 segmentos con nombre `s3://utfsm-inf356-datasets/vlt_observations/vlt_observations_XXX.csv` para poder disponer de un acceso fraccionado a los datos.

El proceso ETL debe procesar el dataset entregado y generar un nuevo dataset para procesamiento posterior, el que debe considerar las siguientes columnas:

- Right ascension - grados
- Right ascension - minutos
- Right ascension - segundos
- Declination - grados
- Declination - minutos
- Declination - segundos
- Instrument
- Exposition time
- Template start (en tiempo unix)

El dataset procesado sólo debe tener los registros que corresponden a las observaciones con categoría SCIENCE y tipo de observación OBJECT (referencia `https://archive.eso.org/eso/eso_archive_help.html`).

El resultado del dataset debe ser guardado en formato parquet en la raíz de su bucket de desarrollo bajo el nombre **vlt_observations_etl.parquet**. Se solicitará incluir el código bajo el nombre code-005.py en su entrega. Describa el procedimiento y muestre alguna métrica relativa a la ejecución y/o resultado de su procedimiento.

```python
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.getOrCreate()

# Stop Spark session
spark.stop()
```

Código 5: Código utilizado para el procedimiento de ETL

3.2. Coordenadas galácticas

Desarrolle un programa que lea el archivo generado por el proceso ETL y genere un nuevo archivo parquet en el que se han transformado las coordenadas RA-DEC ecuatoriales de las observaciones a coordenadas galácticas. El nuevo archivo debe estar en la raíz de su bucket de desarrollo y debe ser llamado **vlt_observations_gc.parquet**. El archivo debe tener las columnas:

- Galactic right ascension (float en grados)
- Galactic declination (float en grados)
- Instrument
- Exposition time
- Template start (en tiempo unix)

Se solicitará incluir el código bajo el nombre code-006.py en su entrega. Describa el procedimiento y muestre alguna métrica relativa a la ejecución y/o resultado de su procedimiento.

```python
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.getOrCreate()

# Stop Spark session
spark.stop()
```

Código 6: Código utilizado para el cálculo de coordenadas galácticas

3.3. Segmentación temporal

Segmente el archivo de coordenadas galácticas en tantos archivos como sea necesario de modo de contar con una agrupación por año y por semana del año. Se considera la primera semana del año la semana del primer lunes de un año. Los primeros días del año pertenecen al año anterior si ocurren antes del primer lunes del año.

Los datos deben ser guardados en formato parquet en una carpeta llamada **partition** en la raiz de su bucket de desarrollo. Dentro de partition las carpetas se separarán por año. Dentro de la carpeta de un determinado año deberá haber un archivo denominado **vlt_observations_XXXX.parquet** con todos los datos del año, y una carpeta denominada weeks, que dentro debe tener una serie de archivos denominados **vlt_observations_XXXX_YY.parquet**, donde XXXX corresponde al año y YY al número de semana comenzando en 0.

Se solicitará incluir el código bajo el nombre code-007.py en su entrega. Describa el procedimiento y muestre alguna métrica relativa a la ejecución y/o resultado de su procedimiento.

```python
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.getOrCreate()

# Stop Spark session
spark.stop()
```

Código 7: Código utilizado para segmentación temporal de datos

3.4. Conteo de observaciones

Desarrolle un programa que reciba un archivo parquet con el formato previamente utilizado para las coordenadas galácticas y que genere un conteo de las observaciones para cada segmento de 10 grados horizontal y vertical del cielo, segmentado por cada instrumento. El archivo de salida debe tener el mismo nombre que el archivo de entrada pero con un .count antes de la extensión .parquet. El conteo tiene que tener como coordenada de referencia el centro de la región angular. El tiempo de exposición debe ser reemplazado por el tiempo total de observaciones que han sido considerados en la cuenta. Se debe descartar la columna con el tiempo de inicio del template.

Se solicitará incluir el código bajo el nombre code-008.py en su entrega. Describa el procedimiento y muestre alguna métrica relativa a la ejecución y/o resultado de su procedimiento.

```
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder.getOrCreate()

# Stop Spark session
spark.stop()
```

Código 8: Código utilizado para segmentación temporal de datos