**PySpark en local (Pràctica)**

# Creació d’un ResilientDistributedDataset o RDD des d’una llista

1. Obrim Jupyter i completem el següent scrip, on el RDD es dirà myRDD:

*# Import PySpark*

**from pyspark.sql import SparkSession**

*#Create SparkSession*

**spark = SparkSession.builder.appName('ElNomDeLaMevaApp').getOrCreate()**

*# Completa-ho, per exemple a ‘ASIX’ hi ha 20, ‘SMX’ hi ha19, ‘DAM’ hi ha 17 i ‘DAW’ hi ha 21, pista (Python List of arrays)*

**myRDD= sc.parallelize**(....)

*# Visualitza-ho*

**myRDD.take(4)**

2. Pregunta: Què significa «sc» davant del mètode «parallelize»?

és el mètode paral·lelitzat de SparkContext per crear una col·lecció paral·lelitzada. Això permet a Spark distribuir les dades entre diversos nodes, en lloc de dependre d'un sol node per processar les dades.

3. Utilitzant aquest métode és pot distribuir myRDD en 4 nodes? Aquests 4 nodes operarien en paral·lel?

Si, Cada conjunt de dades en RDD es divideix en particions lògiques, que es poden calcular en diferents nodes del clúster.

4. Quin perill, en relació als «workers» i el «driver», té usar take() sense especificar el valor n en un RDD molt gran? Quines diferències hi ha amb el mètode «collect»?

# Collect mostra el contingut i l'estructura/metadades.ex.

# Take es pot utilitzar per mostrar contingut i estructura/metadades per a un nombre limitat de files per a un conjunt de dades molt gran. tingueu en compte que aplana les dades i es mostra en una sola fila.

# Creació d’un ResilientDistributedDataset o RDD des d’un arxiu

1. Obrim Jupyter i completem el següent scrip, on el RDD es dirà myRDD i llegirem un arxiu TXT o CSV que tindrem en local (ruta relativa o absoluta) i el text podrà estar separat per coma o per tabulador.

*# Import PySpark*

**from pyspark.sql import SparkSession**

*#Create SparkSession*

**spark = SparkSession.builder.appName('ElNomDeLaMevaApp').getOrCreate()**

*# Completa els 2 espais proposats*

**myRDD= sc.textFile**(....).**map(lambda element: element.split**(...))

# Visualitza-ho

**myRDD.take(5)**

On Split pertany a pyspark.sql.functions.

2. Si usem el mètode «textFile» passant etiquetes «hdfs:», «s3:», «wasb:», «gs:» o «dbfs:» quin tips d’arxius estariem ingestant enlloc de fitxers locals? Especifíca les 5 respostes.

*HDFS (sistema de archivos Hadoop ) - Se utiliza para escalar un solo clúster de Apache Hadoop a cientos (e incluso miles) de nodos. HDFS es uno de los componentes principales de* [*Apache Hadoop*](https://www.ibm.com/co-es/analytics/hadoop)*, siendo los otros MapReduce y YARN.*

*S3: es compatible con los archivos JSON y CSV si se utilizan los valores s3selectCSV y s3selectJSON para especificar el formato de los datos.es posible que las aplicaciones recuperen únicamente un subconjunto de datos de un objeto. Para Amazon EMR.*

*Wasb: El controlador WASB permite usar una clave de acceso de la cuenta de almacenamiento o una Firma de acceso compartido (SAS).*

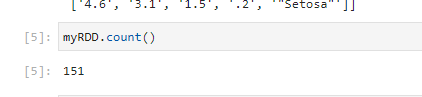
*Gs :la lectura de archivos JSON de Google Cloud Storage (GCS) directamente. Puede usar API similares para leer XML u otro formato de archivo en GCS como marco de datos en Spark.*

*dbfs:es un sistema de archivos distribuido montado en un área de trabajo de Azure Databricks y disponible en los clústeres de Azure Databricks.DBFS es una abstracción del almacenamiento de objetos escalable que asigna llamadas de sistemas de archivos similares a Unix a llamadas API nativas del almacenamiento en la nube.*

3. Si després del «take», afegim al script una línia més com la següent:

**myRDD.count()**

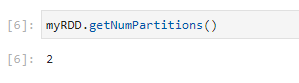
quin resultat estem observant?



4. Si després del «count», afegim al script una línia més com la següent:

**myRDD.getNumPartitions()**

quin resultat estem observant?



5. Quina és la importància de «getNumPartitions()» ó dit d’una altra manera a què ens invita a comprobar?

Devuelve el numero de particiones de RDD.

6. Dins de «textFile» tenim que «minPartitions» és opcional, prova un valors diferent al que has posat en la primera pregunta i/o prova sense informar un valor per a «minPartitions», després comprova-ho amb «getNumPartitions()». Quan creus que és convenient que Spark optimitzi i quan és convenient que tu mateix especifíquis un mínim per al numero de particions?

7. Com més particions més paral·lelisme (sempre, quasi sempre,mai...raona)?

8. Com més paral·lelisme més eficient és la «query» (sempre, quasi sempre,mai...raona)?

9. Dins de «textFile» tenim que «useUnicode» és opcional, si no especifiquem ni «useUnicode=True» ni «useUnicode=False» quina característica pren «textFile» per defecte?

10. Ara treu-li la funció «map()» que està aplicada a «textFile», com canvia els resultats?

11. Dins del codi de «map()», et recorda el terme lambda a les lambda de Python? Què és una funció anònima?

12. Si a «Split» posem « \t» quin és el separador? Pots fer servir el teu nom com a separador, si el tinguéssim al fitxer?

13. Intenta agafar un fitxer molt gran i carrega’l varies vegades amb paràmetres tant del minPartitions com del map() que et donin joc (per defecte, grans, petits,...). La comanda myRDD.count() en algun moment t’informa de Output Duration en l’ordre de magnitud de segons? Et serveix per refinar el teu criteri d’ajust de paràmetres?

14. Per acabar, introduim el mètode «sample» i ho visualitazarem tot en una sola línia, es a dir, ...map(...).sample(...).take(...) «punt map punt sample punt take». El fitxer ha de ser prou gran i experimenta amb tots els paràmetres de «sample», entens quin efecte té cadascun? Quin valor prenen per defecte en cas que siguin opcionals? Quina utilitat té en grans volums de dades saber prendre mostres aleatories? Creus que «sample» et serveix per a les teves tasques d’analisi o hauries de codificar-te la teva propia funció?

15. Dins PySpark hi ha també Filter, Select, Join, Distinct,...et son familiars, pots fer una comparativa amb altres llenguatges o llibreries on els utilitzes?