Buenas prácticas Spark en Datio

Buenas prácticas de Spark en Datio

01 Recursos compartidos: excluyentes y limitados
02 Cache y Persist
03 Dataframe partitions
04 Prueba de procesos pesados
05 Referencias útiles

01

Recursos compartidos: excluyentes y limitados

Asignación de recursos

Etapas de asignación:

Cada usuario tiene unos recursos prefijados que se le asignarán cuando trabaje en Intelligence, en **2 etapas**:

1. Al acceder a Intelligence Tool, dar click en go to tool y (click en Start My Server): se carga la máquina Driver, con la definición de su talla.

2. Al ejecutar la primera acción de Spark: se solicitan los nodos en el pool de recursos para satisfacer la definición de talla.

Los recursos para las tallas pequeñas son más fáciles de conseguir. Para las tallas grandes puede ser más difícil, pues no comienzan las ejecuciones hasta que no se **consiguen todos los recursos** que tienen definidos según su talla.

Asignación de recursos

Observaciones

- Hasta que no se complete la segunda etapa, no se disponibilizan los recursos, y el usuario ve un * en Jupyter.

- Si en 5 minutos no se consiguen los recursos, devuelve **error 504**. Indica que el sistema no tiene recursos disponibles.

- Los recursos se liberan para el resto de usuarios del SB al hacer ***log out***.

02

Cache y Persist

Uso de recursos

Cache y Persist usan recursos de los nodos

Cache y persist usan los **recursos de los nodos**, de manera que puede ser un cuello de botella realizar demasiados cache y persist en máquinas que están pensadas para realizar cálculos.

Para saber si se darán problemas de memoria, hay que comprobar que el tamaño que se quiere cachear/persistir (**volumen del Dataframe**) sea **menor** que la disponible (**recursos de los executors** de Spark).

- Se pueden comprobar los recursos de Spark desde la **SparkUI**

03

Dataframe Partitions

Particiones

Número de particiones de un dataframe o RDD

Spark hace estimaciones y aplica un número de particiones inicial.

Para mejorar la eficiencia de los procesos, a veces hay que ajustar la cantidad de **particiones**, para que se **reparta** mejor la **carga**.

- Muchas particiones: fraccionan demasiado el trabajo. Hará que cada partición tenga menos datos o no tenga datos.

- Pocas particiones: hacen bloques demasiado grandes, que puede impedir la ejecución de las tareas.

Particiones

Número de particiones de un dataframe o RDD

Idealmente, hay que procurar la **menor cantidad** posible de **particiones**, pero con particiones cuyo **tamaño** sea por **debajo del tamaño del bloque** (**128MBs** en DATIO).

Para saber el número de particiones ideal, necesitamos saber aproximadamente el **volumen de datos** que manejamos.

Observación: también hay que tener en cuenta la **configuración del cluster**. Spark puede ejecutar una tarea simultánea para cada partición de un RDD, hasta el número total de cores del cluster.

04

Prueba de procesos pesados

Procesos pesados

Técnicas para trabajar con operaciones pesadas

Antes de realizar operaciones pesadas sobre conjuntos masivos de datos, es mejor **probar con conjuntos pequeños** para comprobar la **viabilidad de las operaciones.** Esto **evita sobrecargar** la plataforma con pruebas y ralentizar el trabajo por motivos de memoria.

- **Limitar la cantidad de muestras**. Por ejemplo, haciendo un *sample* y un *limit.* Estadísticamente, una buena selección de datos lleva a crear un buen modelo.
- **Limitar la complejidad de cálculos,** limitando por ejemplo los parámetros en *paramGrid* (en Random Forest, 20 árboles en lugar de 600 para comenzar).

05

Referencias útiles

Referencias útiles

- **Buenas Prácticas**

https://docs.google.com/document/d/1ofTJJOsPt-h_fCA5RObpJi6V40bAHN6GpssXJMhg8n8/edit

- **Datio en Ether Platform**

https://platform.bbva.com/en-us/developers/introduction-datio/documentation/what-is-datio

- **Jira Datio Service Desk**

https://platform.bbva.com/en-us/disciplines-governance/disciplines/daas/datio-support/documentation/getting-started/welcome-pack

Thanks