**Preguntas típicas para consultor Big Data (spark, hadoop)**

1. En que componentes se divide el framework Hadoop?
   1. Explique cada uno de ellos

HDFS – sistema de almacemiento para grandes archivos

MapReduce – procesa datos

Ecosistema – Herramienta con las que trabajar sobre el

* 1. ¿Qué es el factor de replicación y para qué sirve?

Copia los bloques de datos varias veces para dar tolerancia a fallos, disponibilidad y fiabilidad

1. Diferencia entre un NameNode y un DataNode

NameNode es el nodo principal mientras que DataNode es el esclavo, namenode maneja los metadatos y los datanode y mientras que este ultimo procesa la información

1. ¿Qué es Big Data?

Hace referencia a los grandes volúmenes de datos estructurados o no que manejan las empresas, el análisis de estos para obtener ideas para futuros movimientos estratégicos es de lo que se encarga

1. ¿Cuáles son las V’s que definen Big Data? (Menciona 4)

Volumen, velocidad, visualización y veracidad

1. ¿Qué es Spark?

es un motor para el almacenamiento, procesamiento y análisis de grandes volúmenes de datos

* 1. En que se diferencian Spark con Hadoop MapReduce

Spark es mas rápido (ya que utiliza la memoria), sencillo y tienen componentes específicos para estreaming, machine learning…, spark puede trabajar sobre hadoop

* 1. ¿Qué es un RDD?

Tipo de estructura de datos de spark tolerable a fallos capaz de funcionar en paralelo

* + 1. RDD vs DataFrame vs Dataset

Desde la 2.0 dataframe se englobo dentro de dataset, rdd también estan presentes dentro de dataset, los rdd son para tarea a bajo nivel.

Con dataframe si sucede un error solo se veria al ejecutar el programa mientras que con dataset ya se detectan en la compilación.

* + 1. Existen 2 tipos de operaciones básicas que se pueden realizar con los RDD, menciónalas y define cada una.

Transformación – operación que modificara, filtrara, etc… un rdd

Accion – funciones como count para poder sacar valor estadístico sobre la información

* 1. ¿Qué es una broadcast variable y para que se utilizan?

Variables de solo lectura que pueden ser mantenidas en su maquina y no hay que copiarlas, podemos hacer copias de un archivo en distintos nodos de manera eficiente

* 1. ¿Qué son los accumulators?

Variables compartidas que se crean al realizar una operación asociativa y se utilizan para realizar contadores o sumas

* 1. ¿qué es Data shuffle/shuffling? ¿Por qué ocurre? ¿Cómo evitarlo?

Cuando se realiza una transformación que requiere información de otras particiones, dando lugar a otra partición con dichos datos.

Ocurre cuando los datos se reorganizan entre particiones.

Se puede evitar usando broadcast variables. Puede cargarge en una tabla hash y luego transmitirse a cada ejecutar (solo si el dataset es suficientemente pequeño como para caber en la memoria de un ejecutor)

* 1. Tipos de persistencia de un RDD/DataFrame/DataSet - persist()

Memory\_only – almacena como objetos deserializados java en jvm, si sobra se vuelve a recalcular más tarde cuando se necesiten las partes sobrantes

Memory\_and\_disk – igual que el anterior pero lo que sobre lo almacena en disco

Memory\_only\_ser – guarda objetos serializados java, más rápido pero mayor consumo

Memory\_and\_disk\_ser – igual pero lo que sobra lo lleva al disco

Disk\_only – almacena en disco directamente

Off\_heap

* 1. ¿Qué es checkpointing?

Un mecanismo que almacena la suficiente información para que en caso de fallo se pueda recuperar

* 1. Spark SQL vs DataFrame/DataSet

Spark sql nos permite realizar consultas sql sobre dataframes/datasets previos que podemos transformar en otros con la información filtrada

* 1. Cuantas APIs distintas (por lenguaje de programación) posee Spark

4, scala, java, Python y r

* 1. Si tuvieras que hacer un trabajo de limpieza en ficheros de texto (muy grandes) con que API de spark lo harías y por qué

Scala, por su mezcla entre sencillez y rapidez

* 1. ¿Qué es el Catalyst Optimizer?

Un optimizador basdao en la programación funcional de scala, permite a los desarroladores ampliar y personalizar las funciones del optimizador

* 1. ¿Qué es un ejecutor (executor)?

Procesos del worker node que se comunicacn con el driver y ejecutan tareas.

* 1. ¿Por qué el rendimiento de Spark Scala es mejor que el de PySpark?

Aunque el procesamiento de datos lo realizan los procesos de Python, la jvm de spark sique manejando la persistencia y la transferencia, que se reutilizan desde el paquete central de scala.

1. Tipos de formatos de datos columnares

Amazon redshift, mariadb, cassandra, monet

* 1. Ventajas/Desventajas

Ventajas: posibilidad de compresión debido a que todos los datos de una columna son del mismo tipo, más rápidos a realizar operación de evaluación en grandes volúmenes, menos requisitos.

Desventajas: en aplicaciones transaccionales sería más lenta ya que los datos nuevos se deben distribuir en toda la base de datos.

1. ¿Qué es Hive?

Infraestructura de almacenamiento de datos sobre hadoop para dar consultas, agrupaciones y análisis de datos

* 1. Características

Funciones definidas por el usuario

Consultas con lenguaje sql

Almacenamiento en distintos formatos

Indexación

* 1. ¿Qué es el particionado y el bucketing?

Son maneras de mejorar el rendimiento al realizar consultas eliminando el escaneo de toda la tabla

Particionado divide una tabla en partes relacionadas en función de las columnas

Bucketing divide los datos en archivos más manejables

1. ¿Qué es Impala?

Un motor SQL para operar sobre grandes volúmenes de datos

* 1. ¿En qué se diferencian Hive e Impala?

Ejecuta las consultas directamente sobre el cluster lo que lo hace de entre 5 a 20 veces más rápido, este tiene una latencia mucho más baja y no puede usar índices

* 1. ¿Cuándo utilizarías cada uno de estos?

Hive para introducir los datos e impala para leerlos

1. ¿Qué es sqoop? ¿Para qué se utiliza?

Una aplicación para intercambiar datos entre bases de datos relacionales y hadoop

* 1. Sqoop vs Hive

Una almacena y permite hacer consultar sobre hadoop mientras que la otra puede transferirle bases de datos a hadoop

* 1. Sqoop vs Spark

Una está diseñada para procesar datos y analizarlos y la otra para traspasar datos entre hadoop y bases relacionales

* 1. Sqoop1 vs Sqoop2

Sqoop1 usa un modelo cliente donde necesita instalar sqoop con los conectores y drivers en el cliente, sqoop2 usa un modelo basado en servicio donde los conectores y drivers son instalados en el servidor

Conectores RDBMS: sqoop 1 si, 2 no.

Transferencia de datos de RDBMS a Hive o HBase: sqoop1 si, 2 no

Sqoop2 está en desuso y Cloudera recomienda sqoop1

1. ¿Qué es cloudera?

Una compañía que da servicio de software apache hadoop

1. ¿Qué es Hue? ¿Qué es Ambari?
   1. ¿Para qué se utilizan?

Hue es una interfaz web para la gestión de hadoop

Ambari es una interfaz web para la instalación y administracion de hadoop

1. ¿Qué es una función lambda?

Funciones sin nombres que se definen como una línea que ejecuta una sola expresión

1. ¿Qué es una tupla?

Una fila compuesta de datos en una tabla

1. ¿Qué es programación orientada a objetos?

Programación basada en la creación de clases con las que se crean objetos que se relacionan

* 1. ¿Qué sabes de la programación orientada a objetos?

Es un estilo de programación basada en clases y objetos

* 1. ¿Qué es la herencia de clases? Ejemplos

Las clases hijo heredan los atributos y métodos de la clase padre

Vehiculo: motor, ruedas… una clase hija por ejemplo autobús las heredaría

* 1. ¿Qué es el polimorfismo? Ejemplos

Capacidad de un objeto de dar una respuesta distinta a otro cuando se llama al mismo método

Un método aterrizar() variara en función de si es un helicóptero o un avión

1. ¿Qué es programación funcional?

Es un paradigma de programación declarativa basado en el uso de verdaderas funciones matemáticas. En este estilo de programación las funciones son ciudadanas de primera clase, porque sus expresiones pueden ser asignadas a variables como se haría con cualquier otro valor; además de que pueden crearse funciones de orden superior.

* 1. ¿Qué has usado/aplicado de programación funcional?
  2. Características de la programación funcional.

Transparencia referencial que impide tener efectos colaterales

No existe la asignación de variables

Falta de construcciones estrucuradas

Pueden ser puras o hibridas, estos últimos admiten conceptos tomados de los lenguajes imperativos

* 1. ¿Qué son las high order functions?

Funciones que toman como parametros una function y devuelve una function

* 1. ¿Qué son las monads?

Una interfaz que especifica una forma para los composición de los datos

* 1. ¿Qué es currying?

Es una técnica de transformación de una función, esta función toma multiples argumentos en la función que solo toma uno.

1. ¿Cuáles son las principales características del lenguaje Scala?
   1. ¿Qué es una case class?

Son clases que sirven para modelar datos inmutables de modo que ayudan a definir nuevas instancias con cun inmutabilidad

* 1. ¿Qué son los traits?

Definen tipos de objetos especificando la firma de los métodos admitidos

* 1. ¿Qué es un singleton object y cómo se definen?

Es un patrón de diseño creacional que garantiza que tan solo exista un objeto de su tipo y proporciona un único punto de acceso a él para cualquier otro código.

* 1. ¿Qué es un companion object y cómo se definen?

Un objeto que ha sido declarado en el mismo archivo como clase y tiene el mismo nombre que la clase, puede acceder a los miembros privados de la clase.

* 1. ¿Qué son las clases implícitas (implicit classes)?

Una clase marcada con la clave implicit, esto hace que el constructor principal de la clase está disponible para conversiones implícitas

1. Diferencia entre Clase y Objeto (indistintamente de lenguaje de programación)

Una clase es una estructura de programación definida por el usuario mientras que un objeto es un tipo de dato que surge a partir de la clase programación

Las clases se crean en el proceso de desarrollo mientras que los objetos en tiempo de ejecución

En una clase de definen datos comunes a todos los objetos, luego los objetos son creados utilizando esa información

1. ¿Qué es Jenkins y para qué se utiliza?

Es un servidor de automatización para nuestro flujo de trabajo de código abierto, su base son las tareas, donde indicamos que es lo que hay que hacer en un build.

1. ¿Qué es Git y para que se utiliza?

Sistema de control de versiones distribuido, para subir los proyectos o software y tener siempre a mano

* 1. ¿Qué es una rama?

Un apuntador móvil apuntando a un commit, que nos permite trabajar en un proyecto sin borrar el conjunto de archivos originales

* 1. ¿Cuáles son las principales que crearías para tu proyecto?
  2. ¿Qué es un pull request/merge request?

Pull request: es una petición que el propietario de un fork de un repositorio hace al propietario del repositorio original para que incorpore los commits que estan en el fork

* 1. ¿Cómo trabajas en Git? R: <https://nvie.com/posts/a-successful-git-branching-model/>