**Respuestas a Preguntas Tipo de Examen.**

**1. Según su opinión qué problemas de su vida laboral podrían ser resuelto usando técnicas y/o herramientas de Big Data.**

Los problemas de la vida laboral que podrían ser resueltos usando técnicas y/o herramientas de Big Data serían: el lidiar con la gran cantidad de datos que se recopilan u obtienen de distintas fuentes en nuestra organización, en otras palabras: buscar y encontrar datos de valor, asi como su almacenamiento, observación y procesamiento para obtener información relevante y utilizarlos para identificar nuevas oportunidades de negocios, velocidad en la toma de decisiones, operaciones más eficientes, mayores ganancias y hasta clientes más satisfechos.

**2. Indicar el significado de cada una de las 3 Vs (principales) y explicar cómo es que influyen cada una a la hora de plantear que los problemas de Big Data están en una nueva categoría de problemas de manejo de información.**

***Significado de cada una de las 3 Vs:***

\* ***Volumen***: Los datos crecen de manera exponencial.

\* ***Velocidad***: Los datos se generan con una mayor velocidad y los requerimientos de respuesta son más “real time”.

\* ***Variedad***: Se requieren procesar datos semi-estructurados no estructurados.

***Como influyen cada una a la hora de plantear los problemas de Big Data:***

\* ***Volumen***: Influye en que pueden manejar gran cantidad de datos (terabytes en adelante), de forma rápida eficiente y escalable.

\* ***Velocidad***: Influyen en que los datos necesitan ser obtenidos, almacenados y procesados en tiempo real. Para los procesos en los que el tiempo resulta fundamental, tales como la detección de fraude en una transacción bancaria, el streaming o la monitorización de un evento en redes sociales, estos tipos de datos deben estudiarse en tiempo real para que resulten útiles para el negocio y se consigan conclusiones efectivas.

\* ***Variedad***: Con el auge del big data, los datos se presentan en nuevos tipos, estos pueden ser datos estructurados y fáciles de gestionar como son las bases de datos, o datos no estructurados y semiestructurados, estos dos últimos datos requieren de una herramienta de Big Data específicas, debido a que el tratamiento de la información es totalmente diferente con respecto a los datos estructurados. Para ello, las empresas necesitan integrar, observar y procesar datos que son recogidos a través de múltiples fuentes de información con herramientas cualificadas, como ser Hadoop, MongoDB, Spark, Scala, etc.

**3. ¿Por qué el acceso a los datos es la principal necesidad para una empresa?**

Por que en la pirámide de Kleppmann sobre necesidades de las compañías, el primer escalón a resolver es el acceso a los datos. Se trata de un requisito inicial que solo cuando es resuelto, se puede seguir con el resto de los escalones.

Además tener acceso a los datos me permite identificar nuevas oportunidades de negocios, velocidad en la toma de decisiones, operaciones más eficientes, mayores ganancias, mayor satisfacción de clientes, hacer predicciones y descubrir tendencias.

**4. Indique cuál es la función de cada una de las 3 capas de la Arquitectura Lambda e indique ejemplos de herramientas que se podrían usar en cada una.**

***Batch Layer***: Capa de procesamiento batch que corre periódicamente tomando los datos acumulados hasta el momento y generando los resultados. Se implementa en herramientas como Hadoop o Spark.

***Real Time Layer***: Capa de procesamiento en streaming corriendo constantemente y calculando el “delta” de los resultados entre la última corrida de la capa batch y “ahora”. Se usan herramientas como Stormy Spark.

***Serving Layer***: Capa de consulta que se encarga de mergear los resultados dados por las dos capas de procesamiento y dar la respuesta definitiva.

**5. Con respecto al procesamiento de grandes volúmenes de información ¿Qué beneficios trae que actualmente el hardware sea barato y hasta se pueda adquirir vía cloud?**

Los beneficios del hardware barato son: que permite realizar el escalamiento horizontal necesario para procesar cada vez más tareas y guardar cada vez más datos. Mediante el escalamiento vertical se organiza que varios servidores o nodos trabajen juntos, repartiéndose partes de una tarea para resolverla. Para el cliente que solicita el servicio es transparente el hecho que haya más de una computadora respondiendo. La distribución geográfica de los nodos, también beneficia a los usuarios al conectarse desde diferentes locaciones. El hecho de utilizar varias computadoras, permite además replicar los datos por seguridad.

**6. ¿Cuáles son las principales características de HDFS?**

HDFS es un sistema de archivos distribuido, escalable y portátil escrito en Java para el framework Hadoop. Está optimizado para alto rendimiento y funciona mejor para leer y escribir archivos de gran tamaño, soporta tolerancia a fallos y está diseñado para correr en commodity hardware.

**7. En HDFS cual es la función de estos componentes:**

***Name Node:***

Es un nodo especial que maneja el sistema.

Maneja el espacio de nombres del file system y regula el acceso a los archivos de los clientes del servicio.

Mantiene el árbol del filesystem y la metadata para los archivos y directorios .

Persiste la información en el disco local como una imagen actualizada periódicamente y un log transaccional para el intermedio.

Sabe del estado de los DataNodes para ver si tiene que reasignar.

Responde los pedidos de los clientes para la ubicar bloques.

***Data Node:***

Guarda los bloques que son parte de los archivos.

Reporta cada 3 segundos (configurable) al NameNode su liveness y la lista de bloques que guardan.

Se comunican entre ellos para pasar los bloques que deben ser replicados entre el main y las replica.

**8. ¿Cuáles son las limitantes de HDFS?**

Si falla se pierde todo (Versiones posteriores fueron solucionando).

Muy malo para random data-access.

Se requiere una base de datos como HBase como complemento para estos casos.

Muchos archivos pequeños degradan la performance.

Los archivos pequeños deberían ser evitados particionando o fusionando cuidadosamente.

**9. Explicar cuál es la finalidad de cada una de las siguientes etapas en Hadoop**

***a. Shuffle:***

La operación Shuffle es un paso “lógico” intermedio donde se toman los valores emitidos por los mappers y se los agrupa por clave. De manera que todos los valores emitidos para la misma clave quedan juntos.

[k2 , v2 ] —> (k2 , [vx ,v y ,vz ,...] )

Una tarea adicional del sort/shuffle es retener dichos valores hasta que exista un reducer libre para poder procesarlos.

***b. Reduce:***

La operación reduce la escribe el usuario/desarrollador. Se ejecuta un Reducer por cada clave diferente generada y recibe para la clave asignada todos los valores que generaron los mappers para esa clave.

A partir de la esos valores genera un “valor final” de respuesta para la clave ingresada.

Aunque no es necesariamente siempre se puede pensar como una función de agregación sobre dichos valores.

***Algunos ejemplos “simples” son:***

***Sum***: Realizar la suma de todos los valores

***Count***: Devolver la cantidad (o cantidad de caracteres)

***Identity***: devolver la lista de valores sin tocarla.

Pero puede ser tan compleja como el problema lo requiera.

**11. ¿Qué problemas se intentan resolver en Hadoop con la creación de YARN?**

La version 1.X de Hadoop el componente Map Reduce tenía muchas responsabilidades. Además de procesar la información se encargaba del control de nodos, repartir tareas, comunicarse con el cliente. La complejidad del sistema hacia que tuviera fallas cuando la escala era importante.

En la version 2.X de Hadoop se separó un proyecto llamado YARN (Yet Another Resource Negotiator). El mismo se encarga de manejar los recursos de un cluster, controlar nodos y repartir tareas de cualquier tipo (no necesariamente map reduce.

**12. ¿Cuáles son las principales características generales que definen a las Bases NoSQL?**

No usan un modelo relacional para guardar la información.

No usan el lenguaje SQL para obtener la información.

No fuerzan un esquema por lo que se le puede agregar campos diferentes a diferentes valores.

Corren en cluster de hardware barato.

Escalan horizontalmente simple, por diseño.

Eligen nuevas propiedades en vez de las ACID tradicionales.

**13. ¿Cuáles son los 3 componentes del teorema CAP y que indica el teorema de la relación entre ellos para sistemas distribuidos?**

***Consistency***: Indica que todos los nodos devuelven el último dato escrito.

***Availability***: Garantiza que todo request será inmediatamente respondido (puede ser con error)

***Partition tolerance***: El sistema sigue funcionando a pesar de que está particionado y alguno de los nodos falla o pierde información.

El teorema indica que no se pueden tener las 3, se debe elegir 2. Como estamos hablando de herramientas que escalan horizontalmente necesitamos partición. Por lo que hay que elegir entre Consistencia o Disponibilidad. En general (no siempre) las NoSql eligen disponibilidad por lo que eligen aplicar consistencia eventual.

**14. ¿Qué es lo que se indica al decir que una base cumple con consistencia eventual?**

Promete que dado una escritura, si no hay una nueva actualización todas las réplicas coincidirán eventualmente en el último valor escrito.

La cantidad de réplicas afecta en la velocidad. Menos réplicas la distribución es más rápida, pero el riesgo de pérdida de datos es mayor.

**15. ¿Las bases de datos Columnares por su manera de guardar son muy eficientes para qué tipo de consultas?**

Una base de datos columnar está optimizada para leer columnas de la información (en contrapartida de leer filas).

El principal uso es para hacer queries analiticas, que se ven beneficiadas por este tipo de guardado.

Reducen mucho los requerimientos de accesos de I/O y la cantidad de data a levantar.

Permiten una mejor compresión, ya que la data por archivos es más homogénea.

I/O se reduce ya que solo se leen las columnas requeridas para resolver la respuesta, en vez de realizar un "full scan".

Se diseñan fácil para escalar horizontalmente en hardware barato.

**16. ¿Cuál es la diferencia entre una base de datos key value y una de tipo documento en cuanto al tratamiento del “valor” (pensando que una documento puede considerarse que el documento es el valor)?**

***Base de Datos Tipo Key-Value:***

Un store key-value es una base de datos "simple' que usa un Vector asociativo (un mapa o diccionario) como modelo de datos, los registros se pueden pensar como un par clave – valor. La clave generalmente es un string arbitrario que sirve para identificar al dato y el valor puede ser cualquier cosa y no requiere ni tiene esquema o modelo de dato, esto permite que no se necesiten índices para las búsquedas, pero en contrapartida no se pueden realizar búsqueda por valor.

En general, key-value stores no tienen query language. Proveen operaciones simples:

guardar (put)

borrar (del)

obtener el valor de una clave (get).

Las búsqueda son por clave (get(key)), el valor es opaco, no se puede buscar por elementos del valor, esto permite que no se requieren índices complejos.

Tanto que los índices de clave no son estructuras en memoria. Generalmente se "indiza" usando el file system, todo esto provoca que sean rápidos para escritura y fácil de usar.

***Base de Datos Tipo Documento:***

Es un tipo de bases de datos donde los documentos son un conjunto de pares key-values, donde además los values son objetos complejos.

Características:

se puede filtrar por y actualizar componentes del valor.

Los documentos no deben cumplir con un esquema.

Son perfectos para información semi estructurada.

Los documentos son unidades independientes y completas.

La data completa está contigua y desnormalizada en el disco lo que hace fácil de leer.

La lógica de la aplicación es más fácil de codificar.

No es necesario transformar objetos en tablas.

La data no estructurada se escribe fácil.

**17. ¿Por qué HBASE logra tener consistencia de escritura a pesar de ser distribuido?**

*Una base de datos distribuida (BDD) es un conjunto de múltiples bases de datos lógicamente relacionadas las cuales se encuentran distribuidas en diferentes sitios.*

***Caracteríticas que generan consistencia de escritura:***

Cuando se escribe información en una región, tres elementos participan de la escritura.

**HLog**: un Write Ahead Log que guarda las operaciones de manera secuencial e inmutable

***MemStore***: un LSM-Tree en memoria que guarda las últimas inserciones.

***HFiles***: un archivo que se genera cuando el Memstore excede de cierto tamaño para flushear a disco.

Si la cantidad de HFiles crece para una región más allá de un límite se lanza un proceso de compactación que reduce los archivos y su tamaño.

***Caracteríticas que generan consistencia de borrado:***

El proceso de borrado, escribe una operación de "tombstone" en el MemStore. De manera

que al buscar el registro se encuentra la operación de borrado y se devuelve el no existe.

Cuando se ejecuta una compaction de los archivos es que se borra permanentemente los

registros que fueron borrados lógicamente vía tombstones.

**18. ¿Indicar 3 de los casos de uso comunes para la utilización de REDIS?**

Cache.

Rankings (TOP K).

Session Store.

Job Queue.

PubSub.

Cálculo de Unique Visitors.

Real-time Analytics.

**19. ¿Cuál es la diferencia entre una tabla declarada external y una que no fue declarada así en Hive?**

[*Hive*](https://es.acervolima.com/apache-hive/)*se puede usar para administrar datos estructurados en la parte superior de*[*Hadoop*](https://es.acervolima.com/introduccion-a-hadoop/)*. Los datos se almacenan en forma de tabla dentro de una base de datos. En Hive, el usuario puede crear tablas* ***internas*** *y* ***externas*** *para administrar y almacenar datos en una base de datos.*

***Tabla Interna:***

En general, siempre que creamos una tabla dentro de una base de datos en Hive de forma predeterminada, es una tabla interna también llamada tabla administrada . La razón por la que las tablas internas se administran es que el propio Hive administra los metadatos y los datos disponibles dentro de la tabla. Todas las tablas internas de bases de datos creadas en Hive se almacenan de forma predeterminada en el   directorio / user / hive / warehouse del HDFS. Podemos verificar o anular el centro de almacenamiento predeterminado en hive.metastore.warehouse.dir . Cuando se eliminan las tablas internas (tablas administradas), todos sus metadatos y datos de la tabla se eliminan permanentemente del HDFS y no se pueden recuperar.

*Puntos clave para recordar acerca de las tablas internas:*

La tabla interna admite el comando TRUNCATE

Las tablas internas también tienen soporte ACID

Las tablas internas también admiten el almacenamiento en caché de resultados de consultas, lo que significa que puede almacenar el resultado de la consulta de colmena ya ejecutada para consultas posteriores.

Los metadatos y los datos de la tabla se eliminarán tan pronto como se elimine la tabla.

***Tabla Externa:***

Podemos considerar que una tabla externa es un puntero que señala a un origen de datos administrado y almacenado en otro lugar.

Las tablas externas son una manera excelente de administrar datos en Hive, ya que Hive no tiene la propiedad de los datos almacenados dentro de las tablas externas. En caso de que el usuario descarte las tablas externas, solo se eliminarán los metadatos de las tablas y los datos estarán seguros. La palabra clave EXTERNAL en la instrucción CREATE TABLE se usa para crear tablas externas en Hive. También tenemos que mencionar la ubicación de nuestro HDFS de donde toma los datos. Todos los casos de uso en los que los datos que se pueden compartir están disponibles en HDFS para que Hive y otros componentes de Hadoop como Pig también puedan usar los mismos datos.

*Puntos clave para recordar sobre las tablas externas:*

Hive no llevará datos a nuestro almacén

La tabla externa no admite el comando TRUNCATE

Sin soporte para la propiedad de transacción ACID

No admite el almacenamiento en caché de resultados de consultas

Solo se eliminarán los metadatos cuando se elimine la tabla externa

***20. ¿Donde guarda Hive el contenido de las tablas y donde la metadata (que tablas, que columnas, donde se guardó la data, etc)?***

La queries de Hive corren sobre tablas al igual que las RDBMS.

Una tabla es un directorio en HDFS que contiene uno o más archivos con el contenido de la tablas.

El Path por defecto (configurable): ***/user/hive/warehouse/<table\_name>.***

Los datos pueden ser guardados en diversos formatos (texto, csv, binario, etc).

Los metadatos (tablas, campos, locación de las tablas), se guardan en el "metastore". Que se guarda en una RDBMs asociada.

***21. ¿Qué beneficios trae conocer el esquema de la data a la hora de guardarlas y realizar consultas sobre la misma?***

Los datos se organizan en tablas, y las tablas están construidas sobre particiones. Los datos de las tablas pueden ser accedidos vía consultas SQL.  Cada tabla puede dividirse en particiones que determinan cómo se distribuyen los datos en los sub directorios dentro del directorio principal.

***Ventajas:***

Reduce la complejidad de la programación MapReduce al usar HQL como lenguaje de consulta (dialecto de SQL).

Está orientado a aplicaciones de tipo Data Warehouse, con datos estáticos, poco cambiantes y sin requisitos de tiempos de respuesta rápidos.

Permite a los usuarios despreocuparse de en qué formato y dónde se almacenan los datos.

***22. ¿Cuáles son las características principales de los RDDs?***

Es Inmutable: Los datos con los que se crean no se modifican.

Es particionable y se puede distribuir.

Se guarda en memoria (aunque por necesidad se puede guardar parte en disco).

Fault Tolerant: Sus particiones se pueden recalcular.

Provee ejecución lazy: O sea solo corre sus pipeline cuando se invoca una acción.

**23. Cuales son los dos tipos de operaciones que se pueden realizar sobre los RDDs y que provocan cuando se los invoca.**

Los RDDs proveen dos tipos de operación para procesar sus datos:

Transformación: retorna un nuevo RDD. Los valores del RDD se calculan a partir de aplicar la función pasada a la transformación. Son de ejecución lazy lo que implica que al ejecutarse no se realiza el procesamiento de datos si no que solo se define el nuevo RDD.

Acción: Generan un valor o colección de valores finales. No generan un RDD. Son eager lo que indica que al ejecutarse se procesan los datos (y todas las transformaciones definidas para llegar al mismo).

El juego entre lazy y eager permite tener cierto control sobre el tráfico de red y volumen de procesamiento

**24. Indicar las ventajas y desventajas provee el poder cachear (persistir) un RDD.**

Una de las técnicas de optimización que se puede usar son los métodos cache() y persist(). Estos métodos se usan para almacenar un calculo intermedio de un RDD, DataFrame y Dataset para que puedan reutilizarse en acciones posteriores.

***Las ventajas de usar las técnicas de cache() o persist() son:***

**Rentable**: Los cálculos de Spark son muy costosos, por lo que la reutilización de los cálculos se utiliza para ahorrar costes.

**Ahorro de tiempo:** Reutilizar los cálculos repetidos ahorra mucho tiempo.

**Tiempo de ejecución**: Ahorra tiempo de ejecución del trabajo y podemos realizar más trabajos en el mismo clúster.

https://drive.google.com/drive/folders/1suatdzkYH81xOOOUYIEDHjE7RSF-aaKQ