Parcial 1 – Seminario de Aplicación Profesional

30/06/2020 – Norte – 5º A

Docentes: Gastón Weingand – Federico Adrián Cetraro

Alumno: Di Modugno Juan Emanuel

1. El pilar fundamental del ecosistema Hadoop es la utilización de distintos componentes y servicios distribuidos de software para resolver el procesamiento de grandes volúmenes de datos.

La potencia de Hadoop es que parte de la premisa de resolverlo en forma distribuida, para ello propone y utiliza un sistema de archivos distribuido, esta información es clusterizada y accesible a través de servicios orquestados a través de un flujo de trabajo.

El ecosistema Hadoop permite tener control sobre todo el proceso, desde la ingesta de los datos (import) hasta su posterior consumo (export).

Una última cosa por destacar es que cada servicio tiene una responsabilidad única y su alcance está delimitado de forma clara, esto permite armar cada flujo de trabajo como el administrador considere conveniente.

1. Al diseñar procesos de datos distribuidos obtenemos una ventaja crucial al buscar hacer economía de recursos, esto se relaciona a los distintos componentes intervinientes en cada etapa del proceso puesto que cada etapa utiliza los recursos que necesita para llevarse a cabo.

Lo mencionado previamente favorece la escalabilidad y brinda un grado de transparencia en cuanto a lo que ocurre en cada etapa.

1. Los servicios Cloud están optimizados para el trabajo y utilización en lo referido a cuestiones de Big Data, el motivo de esto es simple de comprender puesto que los servicios prestados en cada proveedor presentan una conexión natural con otros servicios haciendo que estos interactúen de manera limpia… un ejemplo de esto es la posibilidad de tener una Base de Datos NoSQL con características escalables y con los resguardos que esto implique a nivel seguridad de datos (backups, etc), esta Base de Datos es accesible por los demás servicios dentro de la plataforma del proveedor cloud y facilita tanto el flujo de datos así como su consumo.

Analizando a un proveedor, podemos tomar a AWS (Amazon Web Services), nos proveen, entre otros, los siguientes servicios.

* **Amazon DynamoDB** 🡪 Un motor de Base de Datos NoSQL que ofrece una performance extremadamente rápida, escalable y confiable.
* **Amazon Elasticsearch Service 🡪** Un motor de búsqueda y análisis de grandes volúmenes de datos presentados como texto completo.
* **Amazon Kinesis Firehose 🡪** Permite hacer distribución en tiempo real de flujos de datos hacia distintos destinos.
* **Amazon Kinesis Streams 🡪** Es una manera de colectar y procesar flujos de datos en tiempo real,
* **Amazon Kinesis Analytics 🡪** Permite procesar flujos de datos en tiempo real utilizando SQL estándar.
* **Amazon Redshift 🡪** Un data warehouse que presenta simpleza y una relación de costo beneficiosa en cuanto al análisis de datos.

1. Un diseño On-Read es aquel en que el *schema* se crea cuando se leen los datos, esto permite alojar datos no estructurados en una base de datos.

Por su parte, un diseño On-Write genera el *schema* previo a guardarlo, esto permite realizar consultas de manera más rápida debido a que no es necesario inferir el *schema* en base a datos.

1. Al hablar de un Schema On-Read nos referimos a tecnologías NoSQL, permiten almacenar datos de manera «cruda», brindan una capa de **abstracción** en cuanto a cómo se consuma pero es costoso al momento de efectuar la lectura.

Por otro lado, un Schema On-Write se relaciona a tecnologías SQL, requiere establecer un modelo de datos estricto, eso podría decirse que resulta costoso al escribir o definir el esquema pero resulta particularmente económico al realizar consultas a la Base de Datos.

1. A través de gRPC se permite la definición y exposición de servicios. Permite la definición de mensajes-respuesta que establecen un «contrato» de mensaje en el cual se definen datos que serán transmitidos por un ***canal***. Otorga un grado de abstracción en cuanto al tipo primitivo de datos presente en cada lenguaje puesto que se definen Protocol Buffers para llevar a cabo lo concerniente a ***serialización***.

Permiten la definición de ***clientes y servidores*** siendo estos agnósticos al entorno donde su contraparte corre, esta relación puede darse de manera unidireccional o bidireccional, así como de manera sincrónica o asincrónica.

Un último detalle es que gRPC utiliza ***HTTP2***, esto le permite disponer de una mayor velocidad dado el paralelismo que provee el protocolo ***HTTP2*** (entre otras cosas…).

1. SignalR nace como respuesta a la necesidad de manejar el estado en tiempo real a nivel cliente-servidor, esto se traduce a un Cliente que se suscribe a un Servidor.

Se genera entre éstos un canal a través del cual, el servidor puede enviar información (***notificaciones - actualizaciones***) *en tiempo real* a cada cliente suscripto haciendo uso de la tecnología *WebSocket****.***

SignalR se encarga entonces de proveer un marco de ***transporte*** puesto que se genera un túnel (open connection) entre cliente y servidor, esto rompe con lo tradicionalmente provisto por el esquema de requests-responses de HTTP clásico en el cual las conexiones se abrían y cerraban on-demand.

1. WebAssembly (WASM) es un formato de código binario que es descargado y ejecutado en el navegador. Surge como alternativa a la ejecución de Javascript en el navegador.

Permitiendo ***reducir el tiempo de descarga*** del código puesto que en el caso de JS se transmiten como texto plano, mientras que WASM se transmite de manera binaria. (

Otra prestación es la reducción completa en lo concerniente a interpretación de código… una vez descargado el código de Javascript, el mismo debe ser interpretado para su posterior ejecución, este tiempo aumenta proporcionalmente al tamaño del archivo en cuestión… en contraste con Javascript, WASM no insume tiempo en interpretar código debido a que el mismo es nativo y se encuentra a bajo nivel, esto ofrece una importante mejora en materia de rendimiento y optimización.

Una de las ventajas que ofrece a su vez es la posibilidad de permitir a los desarrolladores seguir codificando en el lenguaje de su preferencia y utilizando WASM como objetivo de la compilación.

Un detalle no menor por mencionar es la posibilidad de ejecutar WASM fuera de un cliente que corre en el navegador, pudiendo ser ejecutado este también en un servidor.

Práctico

1. Resultados

A close up of a sign

Description automatically generated

Al llevar a cabo los ***Insert*** en la Base de Datos, las mediciones arrojaron que EFCore es más performante.

A screenshot of a cell phone

Description automatically generated

Al llevar a cabo los ***Select*** en la Base de Datos, las mediciones arrojaron que Dapper es más performante, pero no es concluyente dada la escasa diferencia.