Artículo:

Big Data

1. A New paradigm for big data

Hoy en día se genera gran cantidad de datos en cada segundo que pasa, dicha información puede ser estructurada, no estructurada o semi estructurada. Un gran ejemplo son las redes sociales (Twiter, Facebook, Whatsapp, etc) que generan la mayor cantidad de información.

Este gran crecimiento de la data afecta a los negocios tradicionales a enfocarse en desarrollar o adquirir herramientas que les ayude a tener la data y posteriormente analizar para la toma de decisiones sobre el negocio. Esta administración de la información es fundamental para las empresas.

Tradicionalmente el sistema de base de datos procesa información estructura, tecnologías implementadas por Oracle, IMB y Microsoft. Que tienen una limitación al no poder procesar información no estructura que son proporcionadas por las ya mencionadas redes sociales, llamase mensajes de voz, fotos, opiniones, estados, etc.

Este gran procesamiento de volúmenes de datos se define como un sistema de Big Data, fueron desarrollos por empresas pionera Google, Amazon que desarrollaron soluciones de Big Data. Con el avance de la tecnología se han desarrollado Open Source como Hadoop, HBase, MongoDB, Cassandra, RabbitMQ,etc. que facilitan el desarrollo de los ecosistemas.

La escala en sistema de base de datos tradicional es bien rígida a cambios y tiene que hacerse los ajustes según el problema, esto no sucede en plataformas de Big Data porque son escalables y tienen un proceso eficiente de datos a gran escala. Sin embargo cuando Big Data agrega mucha complejidad en la creación de aplicaciones se debe tener cuidado con la carga de los fragmentos creados en cada nodo que están internalizados dentro de la lógica de programación de las herramientas de Big Data.

Estas herramientas de Big Data no son una panacea pero cuando se utiliza de una manera útil puede ser posible producir sistemas escalables para abordar problemas de datos arbitrarios con tolerancia a fallas humanas.

**Las propiedades de un sistema de datos: complejidad**

Robustez y tolerancia a fallos, los sistemas deben poder capturar las posibles fallas, como los sistemas se caen, coherencia en los datos, datos duplicados, la concurrencia,etc.

Lecturas y actualizaciones de baja latencia: la baja latencia depende de las aplicaciones según el requerimiento algunas necesitan actualizaciones para propagarse de inmediato.

Escalabilidad: la arquitectura lambda te permite poder mantener la capacidad de rendimiento aumento datos, y esta se logra agregando más maquinas.

Generalización: puede aceptar una gama de aplicaciones siendo compatibles en funcionalidad.

Extensibilidad: debe poder realizar cambios sin problemas, en la agregación de una funcionalidad.

Consultas a Ad Hoc: ser capaz de realizar consultas a un conjunto de datos de forma arbitraria.

Mantenimiento mínimo: mantener los procesos en funcionamiento y depurar toda falla en la producción. Con la creación de algoritmos y componentes simples.

Debuggability: el sistema debe ser capaz de brindar notificaciones cuando exista alguna falla. La depuración se logra en la arquitectura lambda a través de la naturaleza funcional de la capa de lotes.

**Las arquitecturas tradicionales son netamente incrementales y traen consigo varias complejidades:**

*Complejidad operacional:* la necesidad de que las bases de datos de lectura y escritura realicen compactación en línea, a medida que un índice de disco se agrega y modifica de manera incremental, partes del índice no se utiliza, estos índices no utilizadas ocupan espacio y, finalmente deben reclamarse el espacio no utilizado. Reclamar el espacio es muy costoso es por eso que se hace en grandes cantidades a este proceso se le denomina compactación.

Para la administración de la compactación, se debe programar las compactaciones en cada nodo para que estas no se vean afectadas a la vez, para esto se debe tener una buena capacidad de disco en cada nodo y suficiente capacidad el cluster. Con la arquitectura lambda las bases de datos principales no requieren ninguna compactación en línea.

*Extrema complejidad para lograr consistencia eventual: una* alta disponibilidad compite con propiedad llamada consistencia. Muchas veces un sistema de alta disponibilidad devuelve resultados obsoletos durante la partición de red.

*Falta de Tolerancia a la falla humana:* un sistema incremental modifica el estado que tiene la base de datos y los errores son inevitables. Tener un arquitectura sincronizada puede estar sujeto errores y una no sincronizada nos da la oportunidad de poder validar y corregir antes de cargar en la base de datos.

Solución Totalmente incremental vs Arquitectura Lambda: las dos soluciones se pueden comparar en precisión, latencia y rendimiento, ambas soluciones deben dar precisiones pero el sistema incremental da de 3 a 5 veces peores. La realización de consultas es mucha más costosa, que afecta a la latencia y al rendimiento.

La arquitectura lambda puede producir soluciones con mayor rendimiento, evitando complejidad que si afectan a las arquitecturas incrementales.

El principal objetivo de la arquitectura lambda es construir un sistema de Big Data con una serie de capas, cada capa satisface un subconjunto de las propiedades y se basa en la funcionalidad proporcionas por las capas inferiores. La función que añade la arquitectura lambda es la ecuación de vista de lotes denominada capa de lote. La capa de lote almacena una copia maestra del conjunto de datos y calcula funciones arbitrarias a este conjunto de datos. Hadoop es un ejemplo claro de un sistema de procesamiento por lotes.

Las capas de servicio y lote satisfacen las propiedades de un sistema de Big Data: robustez y tolerancia a fallos, escalabilidad, generalización, extensibilidad, consultas ad hoc, mantenimiento mínimo, etc.

Capa de velocidad: todo lo que hace referencia para tener un sistema completo de datos en tiempo real, esta capa ve solo datos recientes mientras que la capa mira todos los datos a la vez. Actualiza vistas en tiempo real a medida que recibe nuevos datos.

La gran utilidad de la arquitectura lambda es que no necesita vistas en tiempo real ya que los datos pasan a través de la capa de lote y de servicios, a esta propiedad se le llama aislamiento de complejidad.

La complejidad ha llegado a crear algoritmos como el MapReduce que codifica palabras y calcula las frecuencias de las mismas. Las nubes que sirven para almacenar grandes volúmenes de datos. Información abierta de códigos para Big Data, divididos en 5 categorías: sistemas de computación por lotes, marcos de serialización, bases de datos NoSql de acceso aleatorio, mensajería / sistemas de colas y sistemas de computación en tiempo real.

La importancia de la utilización de ecosistemas de Big Data es porque brinda la posibilidad de disponer de la data para poder construir modelos estadísticos, matemáticos y machine learning para predecir eventos que ayude a la toma de decisiones y al desarrollo de las organizaciones.

Referencias:

Big Data

Principles and best practices of scalable realtime data systems

Nathan Marz and James Warren

Referencias

Big Data

Principles and best practices of scalable realtime data systems

Nathan Marz and James Warren

1. Keystone Real-time stream processing platform

Es la red troncal de datos de Netflix y una pieza esencial de infraestructura que permite una cultura de ingeniería basada en datos. Y ofrece dos servicios fundamentales:

1. Data Pipeline: La plataforma ofrece dos servicios de producción de transmisión de la data; producen, recopilan, procesan agregan y mueven microservicios casi en tiempo real.
2. Streaming procesing as a service (SPaas) :puede crear y operar aplicaciones personalizadas de procesamiento de flujos administrativos, lo que permite centrar en la lógica de la aplicación.

Desafíos:

\*La escala es muy amplia y el desafío también es por eso que se espera que la plataforma Keystone se amplié mientras los suscriptores a Netflix aumente.

\*Diversos servicios no ofrece esta plataforma: como servicio de enrutamiento paralelos según configuración de usuario. Otro procesamiento de flujo de servicio: de SPaas que tiene poca producción en Netflix pero que resalta por la flexibilidad que tiene para ponerse en marcha y por su tratamiento frente a la latencia.

\*Multi tenencia: permite realizar variados trabajados a la vez según el requerimiento del usuario que van desde la entrega de datos hasta el análisis de datos.

\*Elasticidad: para adaptarse a cambios o sobre carga de consultas por parte del usuario.

\*Resilencia nativa en la nube: opera sus microservicios en la nube, por seguridad y facilidad para adaptarse a cambios.

\*Gastos generales de operación: surge la importancia de detallar el ciclo de vida de los trabajos y automatizar la infraestructura por el costo que se lleva acabo

\*Agilidad: está en plan de desarrollo la agilidad de la plataforma

1. Mentalidad de la plataforma y principios de diseño

\*Habilitación: se habilita la plataforma al usuario para que pueda descubrir soluciones novedosas

\*Bloques de construcción: el apoyo que da el usuario en el uso de la plataforma para que se pueda enfocar en la lógica del negocio, esta trae la facilidad que pueda conectarse con el ecosistema de software de Netflix.

\*Compensaciones sintonizables: las limitaciones pueden ser compensables con latencia frente a duplicados, coherencia frente a disponibilidad, etc.

\*el fracaso como ciudadano de primera clase: el diseño puede verse afectado por la no consideración de algunos aspectos.

\*Separación de preocupaciones:

Entre usuario y plataforma, entre el plano de control y el plano de datos, entre diferentes subcomponentes y Infraestructura de tiempo de ejecución.

1. Nuestro Enfoque:

De una arquitectura de reconciliación declarativa a una de autoservicio, la plataforma coordinara los sub servicios como apoyo al manejo de la herramienta.

\*reconciliación declarativa: un resguardo de los estados objetivos para respaldar los daños o perjuicios que se tuviera de la fuente. Este protocolo es provechoso ya que permite que las operaciones sean idempotentes

\*Herramientas de autoservicio: se brinda herramientas de autoservicio donde el usuario pueda manipular para realizar acciones concretas.

\*Motores de procesamiento de flujo: utilización de apache flink

\*Conectores, operadores gestionados y abstracción de aplicaciones: se suministra al usuario estas opciones para la manipulación y la conexión a otros servicios de plataforma

\*Configuración y despliegue inmutable:

\*Autocuración: las fallas son inevitables sin embargo la herramienta cuenta con una autocuración que facilita la funcionalidad y minimiza el error.

\*Rellenar y rebobinar: Las fallas son inevitables, a veces se puede requerir que el usuario cambie o nuevamente rebobine el trabajo de procesamiento

\*Monitoreo y alerta: El monitor de alerta personalizado ayuda al equipo de plataforma e infraestructura a diagnosticar y monitorear problemas.

\*Confiabilidad y pruebas: el usuario puede estar en constante prueba de la herramienta.

1. Ahora y el futuro

El procesamiento de Keystone tiene una escala mayor de trillones de eventos por día. El equipo asociado ha construido y producido varios casos de uso analítica.

Hacia futuro existe un desafío para abarcar temas de : Esquema, capa de servicio que permita una interaccion de plataforma más flexible, proporcine streaming de SQL, casos de uso de Analytics y Machine Learning , etc.

1. Next Generation Data Warehousing as Santander UK

Santander UK utiliza la tecnología de cloudera y la capacidad superior de desarrollo de software para crear para crear su repositorio de datos y análisis de transmición de la inteligencia que mejore las relaciones con sus clientes.

Su aventura para la construcción de un econsistema de Big Data empezó hace 4 años, adoptando la tecnología de Apache Kafka, teniendo la ambición de revolucionar la experiencia del cliente con el uso de datos en tiempo real y análisis del app para usuarios móviles.

Sus herramientas de Big Data son utilizados para el machine learning y Analytics, que tiene una velocidad apropiada para conocer información del cliente., complementándose con la entrega de eventos en la plataforma como servicio (PaaS) integrada de alta calidad y escalable a través de Apache Kafka.

Utilizan para el almacén de data apache Kafka y Apache Kudu, permitiendo el análisis mas rápido de datos. Cuando se combina con Data Vault 2.0 facilita la ingesta de flujo de datos de Apache Kafka, brindando la capacidad de formular preguntas de “aquí y ahora” sobre el comportamiento de los clientes.

Velocidad de Mercado: Santander UK puede afectar las transformaciones de datos al escalar la plataforma de entrega de eventos elástica, que se basa en Scala Akka y Apache Kafka, lo que permite un enriquecimiento de datos rápido y escalable en tiempo real. Esto permite datos más rápidos y oportunos, decisiones más rápidas y mayor velocidad de comercialización para casos de uso debido a la plataforma y arquitectura reutilizables.

El modelo de contribución permite aprovechar los conjuntos de datos puros que se crean de forma independiente por diferentes unidades de la institución financiera. Y si esos datos son valiosos para esas áreas se tiene la capacidad de incluirlos en el almacen de datos,

Goberanlidad de datos: Santander esta innovando en Cloudera uniendo datos de tansmicion continua,principios y marcos de ingeniería de software avanzados ,para mejorar la experiencia del cliente y bienestar financiero del cliente que a largo plazo le traerían grandes beneficios a la institución financiera.

1. Clasificación de texto en transferencias de dinero: un caso de uso de Apache Spark en producción para la banca

El BBVA tiene un motor de búsqueda a través de una descripción textual de una transferencia bancaria, desarrollado en Spark (atendiendo a más de 5 millones de clientes). El procedimiento de los Data Sciencie para clasificar un gran número de transferencias diarias por su texto generando nuevos desafíos de soluciones algorítmicas y los logros establecidos y aprendizajes.

El desarrollo de un aplicativo para poder estimar sus flujos de ahorros a través de los ingresos y gastos históricos basados en sus transferencias de sus cuentas. Y producir pronósticos de gastos futuros en diferentes productos e identificar patrones recurrentes. El cliente también puede comparar su gasto con otros grupos de clientes. Por ejemplo compararse con personas que tengan el mismo ingreso salarial.

BBVA trabaja con un cluster en ciencia de datos (Hadoop) recibiendo más de 7 millones de transacciones diarias. Algunas transferencias con descripción que representan valiosa información y se van categorizando y encontrando patrones de pago. Esto es posible con el MapReduce que ayuda a categorizar patrones y hallar frecuencias.

https://medium.com/@edwincruz9134/a-new-paradigm-for-big-data-71b4b61993dd