REVISTA TIA

Tecnología, Investigación y Academia -Red Avanzada – RITA
Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología

Arquitecturas orientadas a Streaming: Una evolución necesaria

Streaming Oriented Architectures: A Necessary Evolution

Calderón Sánchez, Edison Smith [1]

Citar este documento:

Calderón Sánchez, Edison Smith. Arquitecturas orientadas a Streaming: Una evolución necesaria. Revista Tecnol.Investig.Academia TIA, ISSN: 2344-8288, 10 (2), pp. 116-135. Bogotá-Colombia.

[1] Ingeniero de Sistemas, Universidad Distrital Francisco José de Caldas, Falabella Financiero, 0000-0003-2722-1017, escalderons@correo.udistrital.edu.co, Colombia

Artículo de reflexión
Fecha de recepción: enero 2022 Fecha de aceptación: marzo 2022
ISSN: 2344 8288 Vol. 10 No. 2

Resumen

En este artículo se realizó una descripción y posteriormente una contrastación entre las arquitecturas cliente-servidor, sus variantes, las arquitecturas de procesamiento de datos con las arquitecturas orientadas a streaming, las cuales se han formulado como respuesta a la necesidad de manejar grandes flujos de datos, pero no solo se limitan a ello.

Se hizo una revisión sobre los elementos de las arquitecturas orientadas a streaming, buscando señalar sus virtudes que son acordes a las necesidades de almacenar, procesar y extraer conocimiento de los crecientes flujos de datos producto de la revolución digital que ha ocurrido en las últimas 2 décadas, asunto que atañe a grandes, medianas y pequeñas empresas.

También se ofreció una reflexión para su entendimiento, no solamente como una herramienta para análisis de datos, sino como un referente base para las arquitecturas de software. Finalmente se comentaron algunos primeros pasos que podrían tomar las pymes y particulares en pro de su implementación.

Palabras Clave: *Arquitectura streaming, Kafka, KSQL, Orientación a streaming, Pymes.*

Abstract

This article describes and contrasts client-server architectures, their variants, and data processing architectures with streaming-oriented architectures, which have been formulated in response to the need to handle large data flows but are not limited to this.

A review of the elements of streaming oriented architectures was made, seeking to point out their virtues that are in accordance with the needs of storing, processing and extracting knowledge from the growing data flows resulting from the digital revolution that has occurred in the last 2 decades, a matter that concerns large, medium and small companies.

A reflection was also offered for its understanding, not only as a tool for data analysis, but also as a base reference for software architectures. Finally, some first steps that SMEs and individuals could take towards its implementation were discussed.

Key Words: *Kafka, KSQL, SMEs, Stream-oriented, Streaming Architecture.*

I. Introducción

Al observar el panorama actual se evidencia que han surgido tecnologías novedosas para la administración del software, los servicios, e incluso para la infraestructura, las cuales han llamado la atención y han generado cierto nivel de confianza en las empresas [1] para aventurarse a implementarlas en nuevos proyectos, e incluso lo suficiente como para motivarlas a reconstruir/mudar sus sistemas tradicionales y sistemas internos a dichas tecnologías, entre ellas la nube.

El uso de la nube, y en general, las TICs han permitido a las pymes competir con grandes empresas, pero se ve también como a pesar de este movimiento, pocas son las empresas, especialmente en países no desarrollados [2] que han explorado las capacidades novedosas que ofrece el hecho de tener a disposición los recursos de la nube, tal vez por costumbrismos en su forma de hacer las cosas y en parte también por falta de conocimiento técnico.

Pensando en ofrecer un vistazo en cuanto a las posibilidades, en este artículo se busca hacer un contraste entre arquitecturas tradicionales y las arquitecturas orientadas a streaming, mostrando estas últimas como un modelo nativo para el manejo de la complejidad que suponen los sistemas distribuidos, que se han hecho necesarios en los últimos años (como se describe en [3]), así como para el tratamiento de la creciente cantidad de datos generados en el entorno globalizado en el que las empresas se desenvuelven en la actualidad.

A su vez se pretende brindar algunas sugerencias de por dónde empezar, para que especialmente las pymes, y por qué no, también personas, puedan dar sus primeros pasos en miras a optar por este estilo de diseño arquitectural alternativo que es compatible desde su filosofía con el crecimiento en la digitalización y generación de datos que está y seguirá ocurriendo en los años venideros [4] y del cual las empresas pueden sacar provecho acoplándose correctamente. Esto se hará a través de un paso a paso simplificado que pretende orientar a aquellas personas y/o empresas interesadas acerca de cómo empezar, desde el enfoque hasta una ligera implementación; también se presenta a KSQL como una herramienta de transición en la medida en que utiliza un lenguaje de administración y consulta semejante a SQL, con el que se considera que los interesados podrían dar un primer paso con mayor seguridad y facilidad.

II. Forma tradicional de hacer las cosas

La evolución del hardware y software nos han llevado de resolver problemas en grandes máquinas de propósito específico a contar con dispositivos repletos de aplicaciones que nos permiten realizar una gran diversidad de acciones al alcance de nuestro bolsillo [5]. Durante esa evolución la necesidad siempre fue la misma, poder procesar una serie de datos a través de nuestra lógica de negocio y generar a su vez una salida de datos.

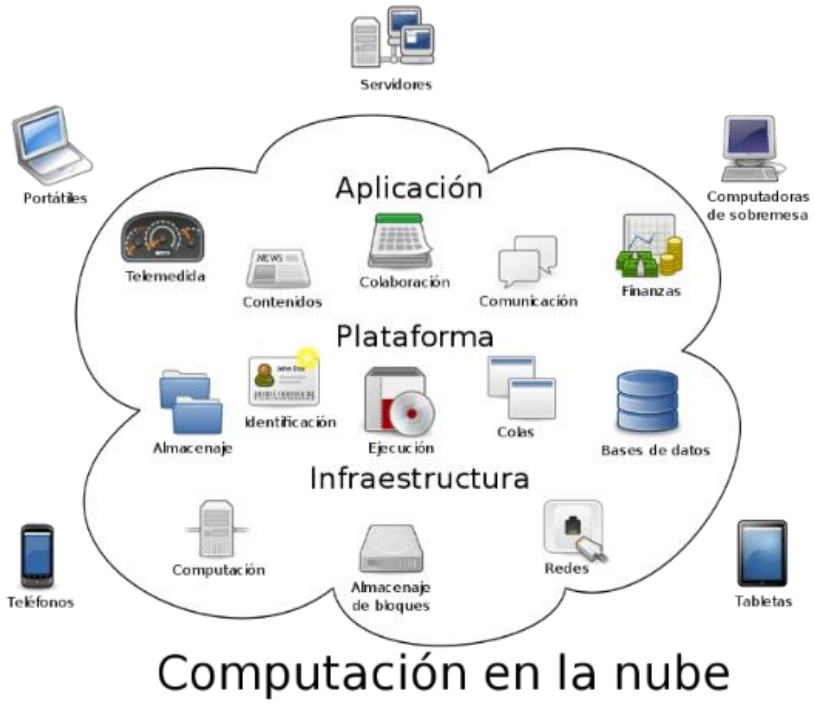
Se pasó de alimentar datos manualmente a un único ordenador, a poder compartirlos conectando ordenadores a través de la red; posteriormente se masificó el uso de ordenadores personales, y más adelante el uso de dispositivos móviles, lo que que día tras día supone la generación masiva de grandes volúmenes de datos.
La analítica de datos siempre ha existido, esto debido a que los datos son un recurso valioso, por ello no es de extrañarse que a la par de la evolución del software también surgieran esfuerzos por definir arquitecturas de procesamiento de estos datos digitales que permitieran a las empresas explotar este recurso.

Así surgieron los modelos de procesamiento en batch donde las empresas tomaban sus grandes almacenes de datos y los procesaban por partes y secuencialmente, cabe comentar que dicho proceso no era rápido y mucho menos económico, y que requería también de capacitación técnica para su administración [6], para ello era necesario contar con grandes servidores que no se podían permitir todas las empresas.

Con la evolución del hardware se fueron abaratando los costos, haciendo posible que más y más empresas montaran sus propios servidores para su funcionamiento interno, para el uso de sus clientes y para procesos de analítica de datos. Sin embargo, como señala [7] no solamente las grandes empresas requieren las virtudes de la analítica de datos, y lo cierto es que de alguna manera las Pymes quedaban excluidas de poder sacar provecho de esta evolución por los costos que suponía.

Quizá el aspecto más revolucionario vino con la posibilidad de llevar todos estos sistemas a la nube, lo que vendría siendo un conjunto masivo de servidores que brindan la posibilidad de acceder a una capacidad de cómputo inmensa a demanda, y que al ser administrados a través de virtualización, tal como lo comentan en [8] permite a empresas y personas economizar costos en la medida en que pueden disponer de un espacio de trabajo, sin tener que administrar los servidores, pudiendo acceder a la capacidad de cómputo que sea requerida según sus necesidades.

Figura 1. Ejemplificación de servicios en la nube como parte del espacio de trabajo de una compañía



Fuente:[9]

Así muchas empresas y personas han visto en la nube una forma económica de llevar sus procesos de cómputo y sistemas de cara al cliente, para que sean accedidos por estos y por la misma empresa a través de internet. Sin embargo, en cuanto a las arquitecturas de software poco han cambiado en los últimos años en el ámbito de las pymes y algunas grandes empresas.

Lo cierto es que son estos mismos proveedores en la nube, y algunas grandes empresas quienes han propuesto algunas alternativas y marcos de trabajo para el desarrollo de ciertas arquitecturas como las que se describen en [10]: orientadas a microservicios, software como servicio, plataformas como servicio, infraestructura como servicio, serverless (computación sin servidor - asignación de recursos a pedido para ejecutar funciones de software) e incluso procesamiento de datos y machine learning como servicio entre otros.

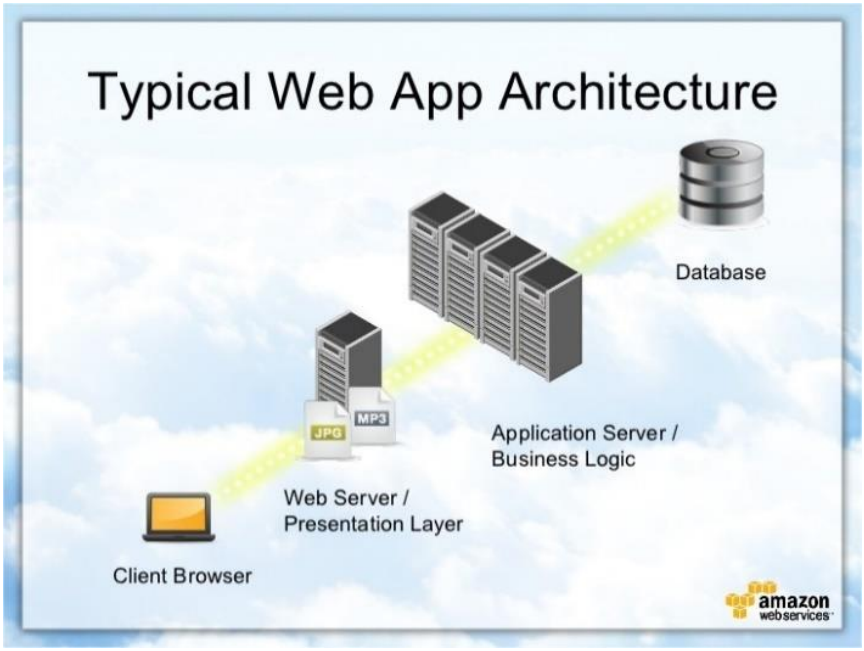
A. Arquitecturas Tradicionales

Una de las grandes revoluciones del software fue poder conectar computadoras, inicialmente entre sí, más adelante computadoras a un servidor. Allí en torno a la década de los 60 surge el modelo de cliente – servidor,

que sobrevive hasta hoy, siendo ampliamente usado en la actualidad, con algunos matices complementarios sobre su definición original.

Una descripción breve de lo que hacen muchas empresas consiste en: Construir un portal web (Frontend - cliente) donde los clientes interactúan, dicho portal brinda sus funcionalidades a partir de un servicio web (Backend -servidor) el cual manipula una única base de datos centralizada donde se consignan los datos del cliente y las interacciones realizadas, y dicha información es administrada por el negocio internamente o en algunas ocasiones a través de un portal administrativo ligado al mismo servicio usado para el portal de clientes. Este modelo tiene ciertas variantes donde se incluyen algunos servicios adicionales, e incluso posiblemente otras bases de datos, sin embargo sigue siendo un modelo tradicional cliente-servidor donde domina la comunicación a nivel de peticiones entre servicios web.

Figura 2. Arquitectura web típica



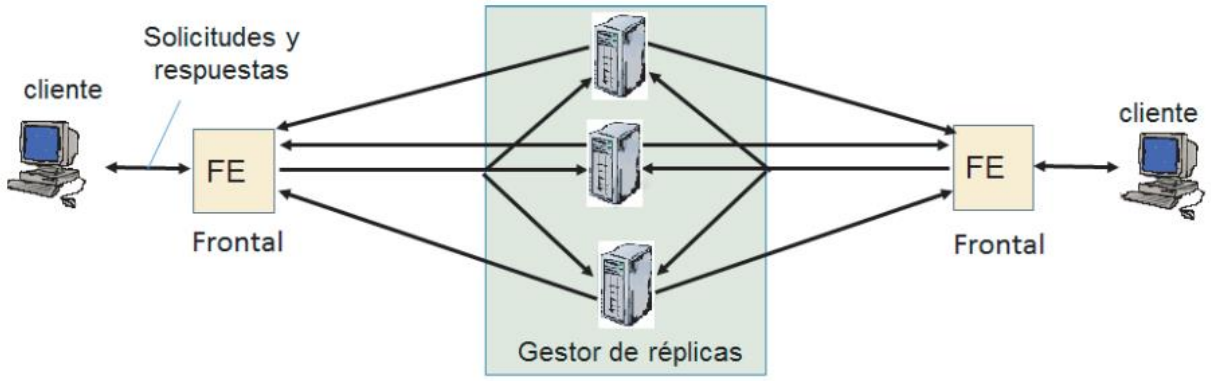
Fuente: [11]

Estos servicios, incluso al estar en la nube no tienen garantía de una disponibilidad del 100%, es por ello que se recomienda replicar los servicios, es decir, subir dos o más réplicas del mismo servicio, para atender las peticiones entrantes; esto por supuesto supone en muchos casos una coordinación para evitar que muchos servicios hagan al mismo tiempo una operación que solo debe realizarse una vez, allí es donde los desarrolladores muchas veces incurren en prácticas indebidas para manejar estos sistemas distribuidos.

B. Sistemas Distribuidos

Los sistemas distribuidos son la respuesta a la necesidad de ofrecer servicios a un gran número de usuarios, ubicados en cualquier parte del mundo y tratar de garantizar que estos cuenten con alta disponibilidad [12]. También fueron la respuesta a la necesidad de procesar grandes volúmenes de datos en tiempo razonable a través de servicios dedicados a las operaciones ETL (extracción, transformación y carga).

Figura 3. Modelo de sistema distribuido y replicado para tolerancia a fallas



Fuente: [12]

C. El modelo de procesamiento de datos tradicional

Muchas empresas encontraron en Apache Hadoop, Apache Spark y otras herramientas semejantes para el tratamiento de datos la respuesta a como procesar sus datos para hacer analítica sobre estos, y con ello apoyar sus procesos de toma de decisiones. Sin embargo este tipo de tecnologías no son siquiera sencillas de configurar, y requiere en muchos casos de la contratación de un proveedor o la consultoría con un experto en el tema para la definición de una arquitectura que permita la exportación de bases de datos de información de la empresa a estos sistemas y la mecánica ETL para el tratamiento de los datos; y con ello llegar a la generación de reportes e informes de inteligencia de negocios.

Este modelo por su nivel de elaboración técnico requiere de expertos que en algunos casos se pueden salir del presupuesto de las pymes - quienes buscan un lugar en el mundo digital y poder sacar provecho de los avances tecnológicos - pero que en muchos casos no cuentan con el conocimiento técnico y/o la capacidad económica de tomar una consultoría para su implementación.

Otro detalle por anotar es que, si bien el proveedor realiza un montaje acorde a las necesidades planteadas por la empresa inicialmente, se hará justamente lo solicitado, y si la empresa no cuenta con un personal técnico especializado que reciba una capacitación al respecto de cómo administrar el sistema, se quedará con una especie de "tecnología desconocida", la cual no podrá alterar acorde a sus necesidades cambiantes.

Pensando justamente en aquellas necesidades cambiantes se propone una arquitectura en streaming como una alternativa donde las virtudes de los sistemas distribuidos son intrínsecas a su naturaleza [4], donde también se añade una capa de tolerancia a fallos, y se presenta un cambio en la manera de hacer las cosas con la que tendrá mucho más sentido la definición del flujo de datos y se contará con la posibilidad de realizar analítica en cada paso o incluso a través de la composición de estos flujos.

III. Arquitectura streaming

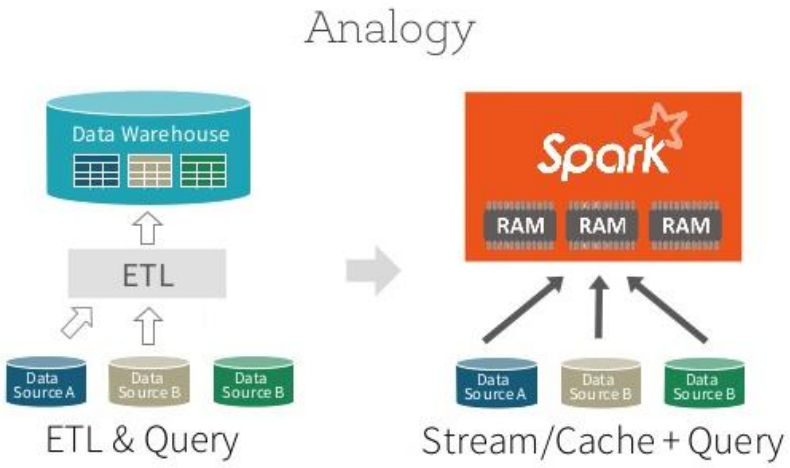
A. ¿Qué es el streaming?

Según [13][14] la proliferación de Internet, la web y la creciente tendencia al uso de dispositivos IoT han alimentado la necesidad de desarrollar aplicaciones que traten los datos como un flujo continuo (stream) en vez de un conjunto fijo. Este es el primer punto de contraste con respecto a las arquitecturas tradicionales, pues en estas cuando se requiere un informe, se acude a una base de datos y se efectúa una consulta o una serie de consultas, proceso que en muchos casos puede ser demorado.

En contraste, en el modelo de procesamiento en streaming se propone que estas operaciones de extracción y transformación ocurran a medida que la información va fluyendo en el sistema, proceso que puede ser llevado a cabo "en segundo plano" sin suponer sobrecarga operativa para el sistema, con la ventaja de poder generar analítica de datos en tiempo real.

La plataforma por excelencia cuando se habla de streaming de eventos es Kafka, claro por sí sola no ayudará a cubrir todos los casos de uso necesarios para una arquitectura, sin embargo se la revisará para entender la filosofía detrás del procesamiento de datos en streaming.

Figura 4. ETL vs SPARK (Plataforma para analítica de datos en tiempo real usando streaming)



Fuente: [15]

B. Kafka

Según su sitio web, Kafka es una plataforma de trasmisión de eventos distribuida de código abierto, utilizada por miles de empresas para canalizaciones de datos de alto rendimiento, análisis de trasmisión, integraciones de datos y aplicaciones de misión crítica [16].

Como bien lo describe [17] Kafka es una plataforma de streaming, que opera bajo el patrón pub/sub, y que internamente usa un sistema de representación de datos en disco (basado en logs), semejante al mecanismo de replicación que utilizan las bases de datos. Este mecanismo además de añadir una capa de persistencia, es el que permite a Kafka hacer su trabajo de manera distribuida a través de las llamadas particiones.

En su arquitectura se observa básicamente una serie de productores quienes son los responsables de la generación de eventos (con su respectiva trama de información, dirigidos a un tópico específico) hacia un conjunto de brokers, donde se realizan tareas de persistencia y distribución hacia consumidores y/o grupo de consumidores.

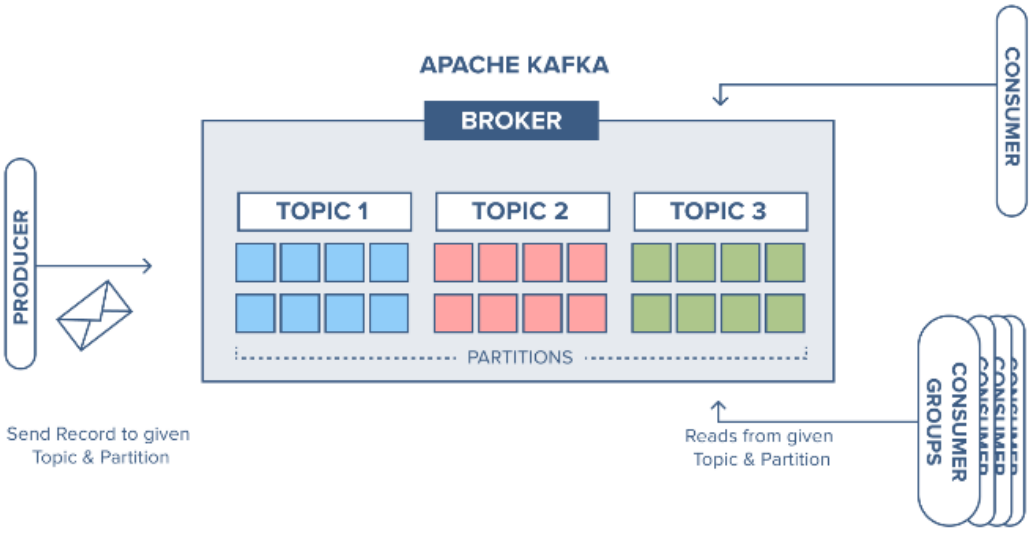
Figura 5. Ejemplo de una trama de datos usando formato JSON


```
{
  "id": 1,
  "first_name": "Rica",
  "last_name": "Blaisdell",
  "email": "rblaisdell0@rambler.ru",
  "gender": "Female",
  "club_status": "bronze",
  "comments": "Universal optimal hierarchy",
  "create_ts": "2018-06-12T11:47:30Z",
  "update_ts": "2018-06-12T11:47:30Z",
  "messagetopic": "asgard.demo.CUSTOMERS",
  "messagesource": "Debezium CDC from MySQL on asgard"
}
```


Fuente: [18]

Internamente Kafka lleva un registro de los grupos de consumidores, de modo que garantice una distribución de los eventos y cuenta además con un mecanismo de confirmación de lectura de los mismos (ACK) pensando en la tolerancia a fallos.

Figura 6. Arquitectura de Kafka y la composición interna del Broker



Fuente: [19]

En la Figura 6 se observa a grandes rasgos la arquitectura de Kafka donde se pueden identificar los siguientes componentes:

- **Productores:**Hace referencia a los orígenes de datos, puede ser una aplicación enviando tramas de información, un sensor enviando sus mediciones e incluso a través de Kafka Connect (componente del ecosistema de Kafka) el envío de un registro completo de las operaciones que suceden al interior de un sistema externo (Ej.: Una base de datos)
- **Broker:**Un clúster de Kafka está compuesto de uno o más servidores denominados Kafka Brokers, los cuales se identifican con un ID entero y contiene ciertas particiones de un tópico. En la ilustración se muestra un solo Broker por simplicidad, pero el número de estos dependerá de la configuración deseada.
- **Tópicos:**Son un nivel de aislamiento que pretende separar tramas que tienen propósitos diferenciados, por ejemplo: Un tópico podría contener tramas relativas a transacciones realizadas por los clientes, otro tópico podría tener tramas relativas a actualización de datos de los clientes, otro podría contener información de los accesos de los clientes a la plataforma; de tal manera que se nota rápidamente un paralelismo de los tópicos con tipos de eventos de un sistema, y cada uno de los mensajes que por el transita con los eventos realizados por un cliente determinado en un momento determinado, el cual tendrá ciertas propiedades intrínsecas a la naturaleza del tipo de evento (ej. Monto, cuenta origen, cuenta destino, etc.).
- **Particiones:**Las particiones son el mecanismo empleado por Kafka para realizar la distribución de carga de trabajo entre los diferentes brokers disponibles. Según la configuración realizada se podrá hacer que todos los eventos se distribuyan de manera uniforme entre los nodos de trabajo disponible, e incluso hacer que toda la información asociada, por ejemplo a un cliente específico, sea procesada por un único nodo (esto a través de un identificador asignado al mensaje, que en el ejemplo del cliente sería su identificador en el negocio).
- **Consumidores:** Los consumidores, al igual que los productores podrán ser múltiples, organizarse en grupos de consumidores (de acuerdo con su propósito, por ejemplo) y también leer contenido de uno o más tópicos, además es posible distribuir la información para que distintos procesos e incluso sistemas hagan una lectura independiente de las mismas tramas para fines diferentes. Los consumidores pueden finalizar el ciclo al procesar el mensaje, o pueden convertirse en productores al enviar el resultado del proceso a otro tópico, que puede ser de interés en otra locación del sistema.

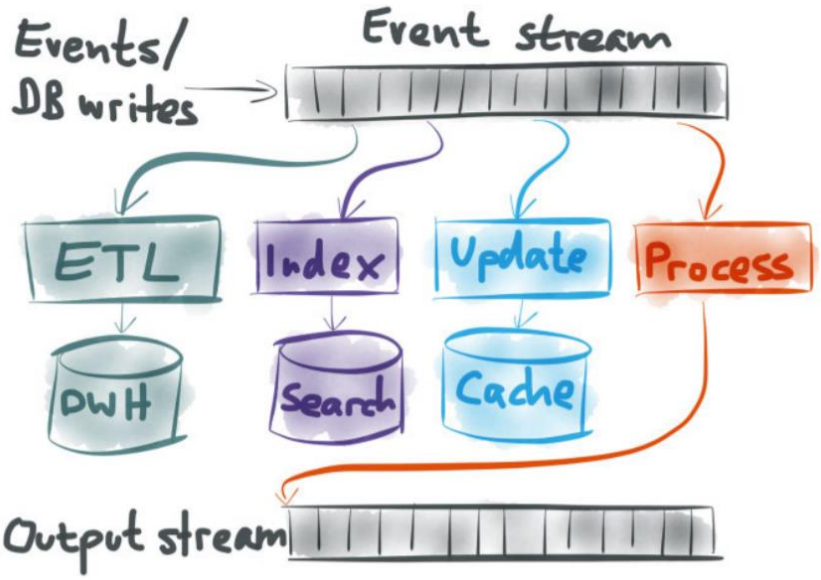
Si se piensa en el sistema transaccional de un banco, este puede tener un grupo de consumidores que se encarguen de guardar los registros de las transacción en una base de datos, otro grupo de consumidores podría

detectar transacciones anómalas en un periodo corto de tiempo y así disparar alertas a un área de fraude, otro grupo de consumidores podría encargarse de notificar al cliente a través de un mensaje de texto sobre la operación realizada y así sucesivamente, se pueden tener tantos procesos como escenarios de negocio existentes.

C. Modelo de procesamiento de datos en streaming

Como se venía anticipando anteriormente, el flujo de trabajo definido en Kafka es completamente compatible con la analítica de datos, en la medida en que la producción de datos ocurre en sí misma a través de la ocurrencia de un evento, de modo que al ver todo como eventos se cambia el paradigma orientado a solicitudes, de tal manera que una acción genera un evento (comunicado por un productor) que puede ser interpretado por distintos interesados (consumidores) quienes procesarán dicha información generando una salida que puede ser una alerta, una operación contra una base de datos o incluso otro evento, generando un nivel de composición con el que es posible definir incluso un flujo de trabajo completo, todo ello a partir de un simple evento.

Figura 7. Muchas posibilidades para un stream de eventos

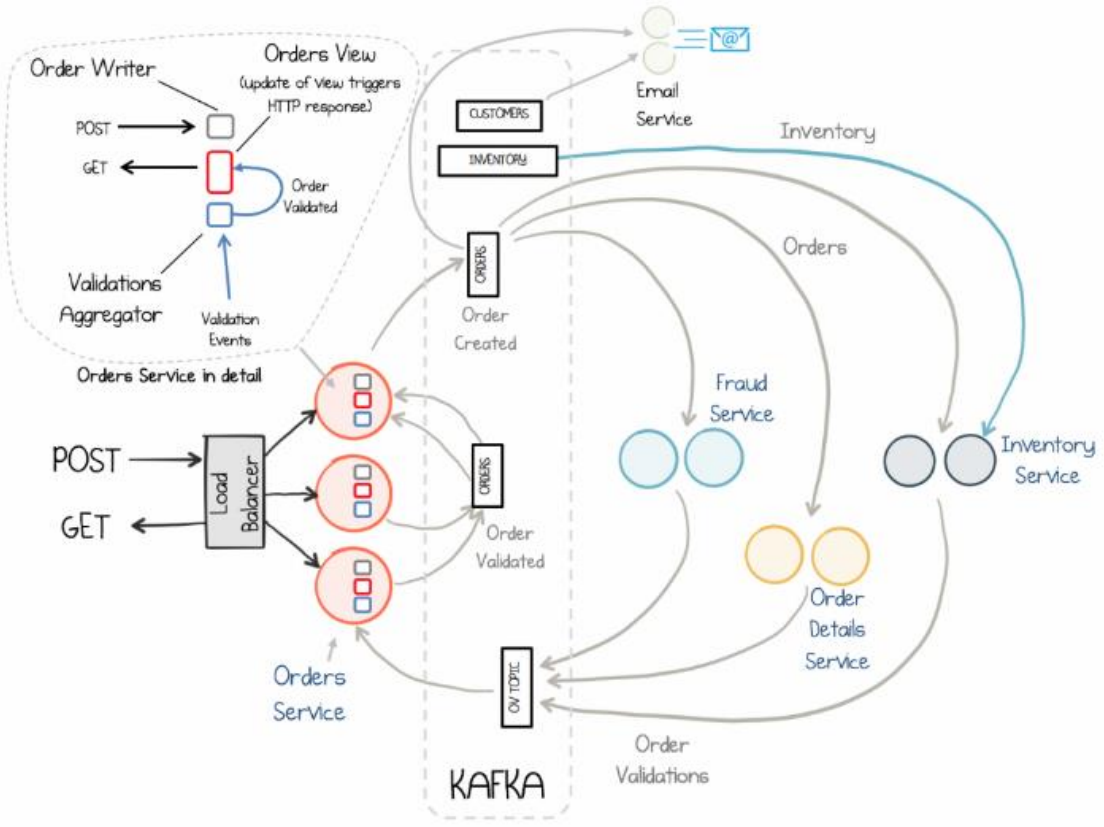


Fuente: [17]

D. Streaming más que procesamiento de datos

Anteriormente se habló de lo adecuado que resultaba el modelo de trabajo en streaming para la analítica de datos, y se ejemplificó a través de Kafka y su filosofía, pero algo interesante que nos plantea [17] es que la arquitectura orientada a streaming permite mucho más que solamente alimentar bases de datos, sistemas de indexación y caché, si no que en realidad se puede usar a Kafka como backend de comunicación entre servicios (en [20] se ofrece un ejemplo de una implementación orientada a streaming) que brinda alta disponibilidad y resiliencia a fallos en la medida en que almacena un registro de los eventos ocurridos durante un periodo de tiempo parametrizable, incluso cuando ya han recibido confirmación de lectura.

Figura 8. Ejemplo de una arquitectura de microservicios orientada a streaming sobre Kafka



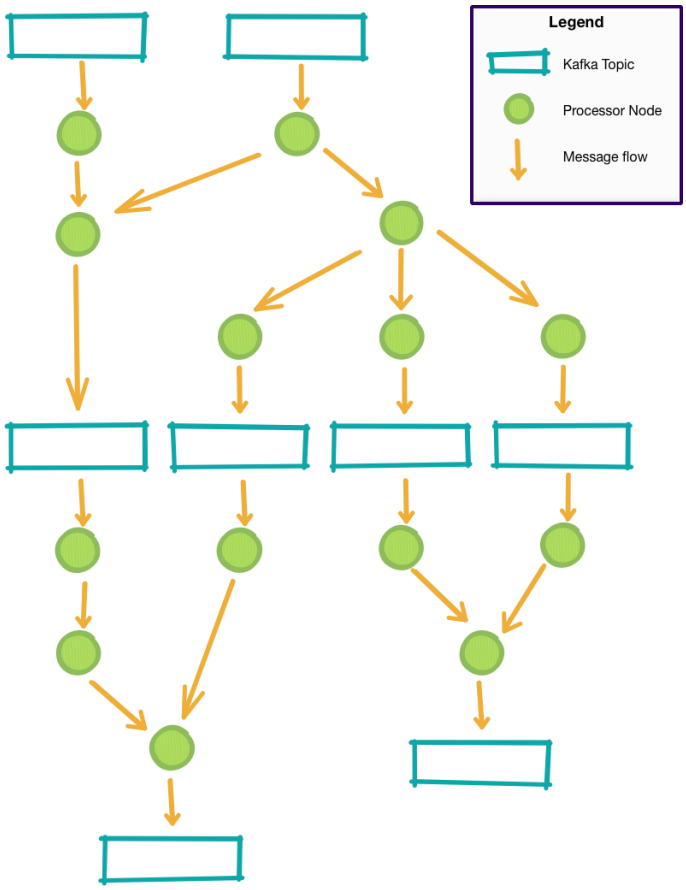
Fuente: [20]

Esto por supuesto permitirá realizar un reprocesamiento de datos en caso de que se deba hacer un ajuste, o en caso de que haya ocurrido un evento adverso.

Pero se debe ser cuidadoso en no acotar nuevamente al streaming como un mero canal de trasmisión de eventos, como en muchas ocasiones se concibe, sino que se debe ver el potencial que supone la posibilidad de definir flujos de datos completos con la filosofía de eventos, tópicos consumidores y productores.

Así se puede definir una especie de flujograma (al mero estilo de un algoritmo), donde en ciertos nodos se definan que transformaciones se harán sobre los datos, y en otros que operaciones se realizarán (notificar, almacenar, analítica de datos) [21].

Figura 9. Topología de procesamiento sobre Kafka Streams



Fuente: [21]

En la Figura 9 se observa el esquema de un flujograma simple, el cual dentro de Kafka Streams (librería para construcción de aplicaciones orientadas a streaming) se conoce como Tipología, y es una forma de definir en los nodos de procesamiento qué operaciones se realizarán, y se tendrá la posibilidad de que dicho nodo se convierta en productor al generar una salida a otro tópico que será procesado por otro nodo de procesamiento.

IV. Como dar los primeros pasos

A. Naturalizando los eventos

El primer paso necesario para asimilar el cambio de paradigma es naturalizar el manejo de eventos. Es una relación natural: cuando un cliente se registra en un sitio, ocurre un evento; cuando alguien visualiza un video, ocurre un evento; cuando alguien hace un retiro en un cajero, ocurre un evento, y si se sabe recolectar la información necesaria asociada al evento se podrá generar una trama autodescriptiva. El siguiente pasó vendrá entonces en la definición de qué acciones se deben tomar en torno a dicho evento, algunas de las que se han descrito en el artículo son: almacenar la información en una base de datos, generar una alerta, realizar un filtrado, monitorear valores anormales o incluso se puede pensar en aplicaciones algo más sofisticadas como revisar tendencias que ocurran en un proceso de tiempo o evaluar un modelo de Machine Learning, todo dependerá de cuál sea la necesidad.

B. KSQL una manera familiar de trabajar en streaming

Previamente se introdujo el concepto de topologías, las cuales pueden ser claras gráficamente pero no necesariamente sencillas de implementar.Pues bien, hay una solución al respecto, la cual es además ideal en la medida en que toma una sintaxis conocida casi que universalmente por cualquier ingeniero de software, desarrollador, programador y demás profesionales de software, y es que KSQL es una base de datos orientada a streaming, la cual según su sitio web [22], es una base de datos diseñada para procesamiento de streaming, la cual opera nativamente sobre Kafka y usa una sintaxis semejante a la que se acostumbra usar en SQL (Structured Query Languaje – Lenguaje de uso extendido en bases de datos transaccionales tradicionales).

Figura 10. Ejemplo de sintaxis usada en KSQ Fuente:[18]

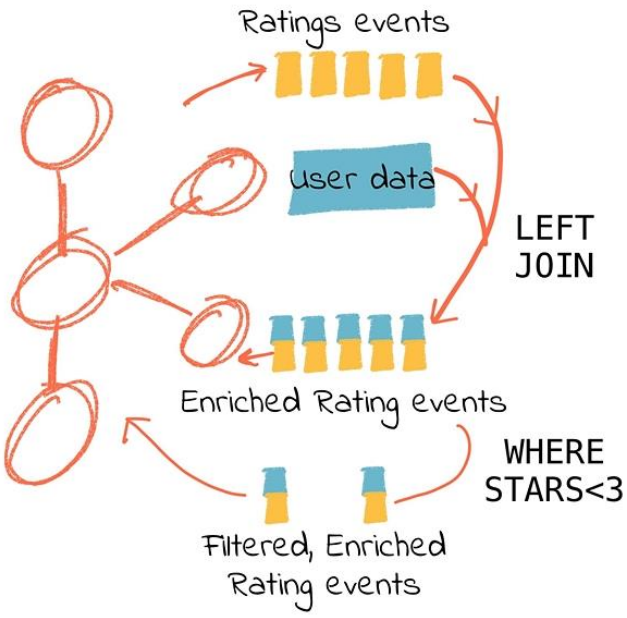
```
-- Process all data that currently exists in topic, as well as future data
SET 'auto.offset.reset' = 'earliest';

-- Declare source stream
CREATE STREAM CUSTOMERS_SRC WITH \
(KAFKA_TOPIC='asgard.demo.CUSTOMERS', VALUE_FORMAT='AVRO');

-- Re-partition on the ID column and set the target topic to
-- match the same number of partitions as the source ratings topic:
CREATE STREAM CUSTOMERS_SRC_REKEY WITH (PARTITIONS=1) AS \
SELECT * FROM CUSTOMERS_SRC PARTITION BY ID;
```

KSQL interpreta las consultas escritas en una sintaxis semejante a SQL, y genera una tipología de Kafka Streams, generando además los tópicos necesarios para el tratamiento de datos que se definan, simplificando y abstrayendo el trabajo que supone dicho andamiaje, ofreciendo una herramienta familiar, incluso en el mismo ámbito de analítica de datos. Este mecanismo es una ayuda para los procesos de analítica en la medida en que permite operaciones join (agregaciones entre datos) de distintos streams basándose en el contenido de los mensajes y en las ventanas de tiempo en que se produjeron, tal como se describe en [18].

Figura 11. Operaciones de join y filtrado en KSQL análogas a SQL



Fuente [18]

El poder contar con un mecanismo familiar debe resultar en un factor de ayuda para las pymes y personas interesadas en dar un vistazo, además de que el poder hacerlo no supone ningún costo puesto que es una base de datos de uso libre.

C. Cómo iniciarse en la práctica

Se ha definido la siguiente guía, paso a paso con el objetivo de que las personas y empresas interesadas en iniciarse con las arquitecturas orientadas a streaming puedan hacerlo por su propia cuenta.

1. Identificar eventos del sistema, y definir atributos de interés con el fin de definir la estructura de la trama.
2. Realizar un montaje de Kafka, para mayor facilidad puede hacerse en la nube, o incluso usando algunos

de los ejemplos completos que Confluent ha preparado [23], donde a través de Docker es posible realizar un montaje completo.

3. Encontrar una librería para conectarse a Kafka en el lenguaje en que se esté trabajando, con el fin de producir un evento con su información inherente. Esto no debería ser un problema puesto que la popularidad de Kafka ha impulsado muchos proyectos de código abierto en diversos lenguajes de programación.
4. Una vez se tengan los flujos de datos en Kafka se puede usar KSQL para realizar las transformaciones y agregaciones de datos que sean necesarias.
5. Para poder realizar alguna acción acorde a los eventos definidos, se debe acudir a los resultados del punto 3 y usar una librería para recibir los eventos, luego de esto se definirá un procesador que se encargará de llevar la acción a cabo. Y listo, se tendrá una implementación sencilla, con las virtudes de los sistemas distribuidos, con una capa de persistencia en caso de que algo salga mal y que puede evolucionar de manera dinámica conforme a las necesidades que surjan.

La guía por supuesto es un paso a paso simplificado que pretende motivar a los interesados a aventurarse a hacer una prueba de concepto, en la práctica se podrían incluir en el ejercicio otras fuentes de datos como pueden ser: sensores, otras bases de datos, sistemas de indexado, sistemas de archivos y otros. Para el procesamiento se podría emplear cualquier herramienta disponible en el entorno de Kafka (KSQL, Kafka Streams, etc.). Y se podrán llevar los datos a bases de datos, sistemas de analítica de datos de propósito específico, servidores de indexado, sistemas de auditoría, dashboards y casi que cualquier sistema que se pueda pensar a través de una integración con Kafka, que puede ser implementada directamente por el interesado o a través de Kafka Connect, donde la comunidad se ha advocado a desarrollar conectores para muchos sistemas de uso común.

V. Conclusiones

A partir del trabajo realizado se puede concluir que si bien las arquitecturas tradicionalmente empleadas son válidas y respaldadas por su amplio uso hasta el día de hoy, vale la pena explorar la alternativa planteada en la medida en que está respaldada por un sistema construido pensando en la alta disponibilidad, resiliencia a fallos y alto rendimiento para aplicaciones de cara al futuro, donde los flujos de información seguirán creciendo y las empresas necesitarán contar con soluciones acordes que les permitan extraer conclusiones en tiempo oportuno para la toma de decisiones y con ello generar y afianzar ventaja competitiva.

De igual manera se espera que la revisión realizada haya sido suficientemente simple para motivar a principiantes en relación con el tema y animarlos a realizar una prueba de concepto que les permita comprobar de primera mano las virtudes expuestas.

También se ofreció una alternativa para abordar el enfoque de una manera sencilla a través de KSQL, como una herramienta de transición del enfoque tradicional al enfoque de orientación a eventos, pretendiendo motivar aún más a los interesados a explorar la temática.

VI. Recomendaciones

A partir de este trabajo se espera motivar a personas y particulares a usar un estilo de diseño arquitectural diferente en sus proyectos, por lo que sería interesante contar con una serie de artículos narrando casos de éxito en la implantación de la orientación a streaming en sus arquitecturas, lo que sirva no solamente de motivador, sino también como una base de aprendizajes y errores comunes.

En este momento me encuentro desarrollando un trabajo conexo donde aplicaré la orientación a streaming como mecanismo para realizar una identificación de patrones en conversaciones con chatbots, las cuales deben generarse en tiempo real, y se hará sobre un sistema existente sin generar impacto en el mismo, únicamente produciendo como evento cada mensaje de la conversación y realizando una evaluación a través de un pipeline de analítica de datos una extracción de tópicos y análisis de sentimientos que permitan generar una infografía sobre la conversación en tiempo real.

Otro trabajo interesante sería una guía un poco más detallada al respecto de la elección de tecnologías, sugerencias de entornos de trabajo y recomendaciones que permitan orientar a los más principiantes a obtener un resultado satisfactorio ahorrando tiempo y esfuerzo.

VII. Referencias

[1] T. Mens, S. Demeyer, M. Wermelinger, R. Hirschfeld, S. Ducasse, y M. Jazayeri, “Challenges in software evolution”, *Int. Work. Princ. Softw. Evol.*, vol. 2005, pp. 13–22, 2005, doi: 10.1109/IWPSE.2005.7.

[2] B. M. R. Wilson, B. Khazaei, y L. Hirsch, “Cloud adoption decision support for SMEs using Analytical Hierarchy Process (AHP)”, *2016 IEEE 4th Work. Adv. Information, Electron. Electr. Eng. AIEEE 2016 - Proc.*, pp. 3–6, 2017, doi: 10.1109/AIEEE.2016.7821809.

[3] P. Dobbelaere y K. S. Esmaili, “Industry paper: Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations”, *DEBS 2017 - Proc. 11th ACM Int. Conf. Distrib. Event-Based Syst.*, pp. 227–238, 2017, doi: 10.1145/3093742.3093908.

[4] L. Querzoni y N. Rivetti, “Tutorial: Data streaming and its application to stream processing”, *DEBS 2017 - Proc. 11th ACM Int. Conf. Distrib. Event-Based Syst.*, pp. 15–18, 2017, doi: 10.1145/3093742.3095108.

[5] P. E. Ceruzzi, “Historia de la informática”, en *Fronteras del Conocimiento*, BBVA, 2008, pp. 109–127.

[6] J. Gomar, “Qué es el procesamiento batch o por lotes”, 2018. https://www.profesionalreview.com/2018/11/25/que-es-el-procesamiento-batch/ (consultado may 21, 2021).

[7] J. Serrano, “Big data y analítica web. Estudiar las corrientes y pescar en un océano de datos”, *El Prof. la Inf.*, vol. 23, núm. 6, pp. 561–566, 2014, [En línea]. Disponible en: http://recyt.fecyt.es/index.php/EPI/article/view/epi.2014.nov.01.

[8] C. Henriquez, J. F. Del Vecchio, y F. J. Paternina, “La computación en la nube: un modelo para el desarrollo de las empresas”, *Prospectiva*, vol. 13, núm. 2, p. 81, 2015, doi: 10.15665/rp.v13i2.490.

[9] Wikipedia, “Computación en la nube”. https://es.wikipedia.org/wiki/Computación_en_la_nube (consultado may 20, 2021).

[10] O. Avila Mejía, “Computación en la nube”, *Ing. Eléctrica. UAM-I*, pp. 45–52, 2011, [En línea]. Disponible en: https://www.mendeley.com/viewer/?fileId=7ffb95a7-1147-0773-1fa1-aab1cd9f2ec6&documentId=33b2eb34-54eb-33a9-93c4-5ec33163bd1b.

[11] A. W. Services, “AWS Architecting In The Cloud”, 2009. https://es2.slideshare.net/AmazonWebServices/aws-architecting-in-the-cloud (consultado may 28, 2021).

[12] F. de A. López Fuentes, “Capítulo 1. Introducción a los Sistemas Distribuidos”, en *Sistemas Distribuidos*, México D.F.: Universidad Autónoma Metropolitana, 2015, pp. 15–21.

[13] S. Chandrasekaran y M. J. Franklin, “Streaming Queries over Streaming Data”, *VLDB ’02 Proc. 28th Int. Conf. Very Large Databases*, pp. 203–214, 2002, doi: 10.1016/b978-155860869-6/50026-3.

[14] T. Dunning y E. Friedman, “Why Stream?”, en *Streaming Architecture*, 1st ed., O’Reilly Media, 2016, p. 119.

[15] Á. Rayón, “TECNOLOGÍAS DE INGESTA DE DATOS EN PROYECTOS «BIG DATA» EN TIEMPO REAL”, 2016. https://blogs.deusto.es/bigdata/tecnologias-de-ingesta-de-datos-en-proyectos-big-data/ (consultado may 28, 2021).

[16] Apache, “Kafka”. https://kafka.apache.org/ (consultado may 23, 2021).

[17] M. Kleppmann, *Making Sense of Stream Processing: The Philosophy Behind Apache Kafka and Scalable Stream Data Platforms*. O’Reilly Media, Inc., 2016.

[18] R. Moffat, “Democratizing Stream Processing with Apache Kafka® and KSQL - Part 2”, *InfoQ*, 2018.

[19] L. Johansson, “Part 1: Apache Kafka for beginners - What is Apache Kafka?”, 2020. https://www.cloudkarafka.com/blog/part1-kafka-for-beginners-what-is-apache-kafka.html (consultado may 24, 2021).

[20] B. Stopford, “Building a Microservices Ecosystem with Kafka Streams and KSQL”, 2017. https://www.confluent.io/blog/building-a-microservices-ecosystem-with-kafka-streams-and-ksql/ (consultado may 26, 2021).

[21] A. Bryant, “Kafka Streams work allocation”, 2018. https://medium.com/@andy.bryant/kafka-streams-work-allocation-4f31c24753cc (consultado may 24, 2021).

[22] Confluent, “KsqlDB”. https://ksqldb.io/ (consultado may 27, 2021).

[23] KsqlDB, “Synopsis, end-to-end examples”, 2021. https://docs.ksqldb.io/en/latest/tutorials/ (consultado may 24, 2021).

Publicación Facultad de Ingeniería y Red de Investigaciones de Tecnología Avanzada – RITA

REVISTA

TIA