Arquitectura Distribuida y Escalable Para Lectura, Procesamiento y Manejo de Información: Enfocado En Un Sistema De Notificaciones Masivas

Camilo Rincón
Escuela colombiana de Ingeniería
Julio Gravito
Bogotá D.C, Colombia
ivan.rincons@mail.escuelaing.edu.co

***Resumen*— En este trabajo se busca analizar y proponer un sistema capaz de realizar la lectura, procesamiento y manipulación de datos masivos de manera eficiente, centrándonos en el contexto de notificación de facturas clientes, se busca orientar este trabajo a utilizar arquitecturas distribuidas actuales tales como microservicios, buscando así una escalabilidad del sistema al momento de procesar grandes cantidades de datos**

Palabras clave: Arquitectura distribuida, microservicios, escalabilidad

I. INTRODUCCIÓN

En los últimos años, la sociedad ha avanzado tecnológicamente a un ritmo muy acelerado en diferentes áreas de la información, como pueden ser: computación en la nube, internet de las cosas (IoT), ciencia de datos, inteligencia artificial, entre otros. Estos avances, han causado de manera implícita, la generación y uso de grandes volúmenes de datos, de hecho, para sustentar esta percepción de crecimiento masivo de la información en los últimos años, nos apoyaremos en un artículo recopilado por *statista*, la cual es una plataforma que recopila información de datos estadísticos de varios sectores en base a estudios de mercado, encuestas y datos gubernamentales. De acuerdo con el artículo el ***Big Bang del Big Data [1]*** el volumen de los datos digitales generados a nivel mundial ha presentado un incremento de hasta treinta veces mayor al crecimiento promedio de los ultimo diez años. En el cual también menciona que la proyección de este crecimiento continuara, planteando que en términos de almacenamiento llegaremos hasta los 180 zeta bytes de información únicamente para el año 2025, como se muestra en la *Fig. 1.*

A raíz de esto ha surgido la necesidad de contar con sistemas con mayor eficiencia y capacidad de procesamiento, dado el desbordado de volumen de usuarios y de datos en el mundo antes mencionado, lo que ha causado que, en la actualidad, el rendimiento de los sistemas al momento de procesar se convirtiera en un aspecto de suma importancia, es tanto así la creciente de los volúmenes, que estos han logrado impactar desde el usuario individual, hasta las grandes empresas, donde estas últimas han tomado mayor relevancia ya que estas pueden ser afectadas de forma económica. Un ejemplo que podemos plantear es el de algún banco o entidad financiera que puede llegar a perder miles, e incluso millones de dólares si sus sistemas no logran atender los grandes volúmenes de información o de usuarios en un día. Donde la no generación de las transacciones no serán la única consecuencia en términos monetarios, ya que también incluye, perdidas de clientes, o incluso multas y sanciones, como fue el caso que ocurrió aquí en Colombia **¡Error! No se encuentra el origen de la referencia.** con Bancolombia donde nos cuenta que, esta entidad financiera sufrió una sanción de hasta ochocientos cuarenta millones de pesos, por fallas técnicas, que causaron la negación del servicio a los clientes.

Es por esto que los sistemas deben contar con características tales como una alta disponibilidad, alta escalabilidad y alto rendimiento dado el aumento de datos y/o solicitudes como las que se han venido sufriendo en los últimos años, complementando el problema de la información en la actualidad, existe también la problemática que aún se pueden encontrar sistemas con arquitecturas de manera centralizada, con un único bloque lógico conocido como *arquitecturas monolíticas*, ya sea porque en sus inicios era una arquitectura rápida y útil pero a medida que empezó a tener la necesidad de aumentar su capacidad de soporte al volumen de usuarios/datos, empezaron a existir problemas, los cuales se buscan solucionar con arquitecturas más modernas.

Para entender las diferentes estrategias y arquitecturas empleadas en la actualidad, se realizaron búsquedas en Google académico y Google, encontrando varios artículos de interés para el desarrollo de la presente investigación. Por lo tanto, a continuación, se presentan algunas arquitecturas, frameworks de trabajo y metodologías que ayudan a procesar datos masivos de manera distribuida, manteniendo un estado del sistema consistente y fácil de mantener ante un cambio.

- Arquitectura orientada a microservicios
- Spring Cloud
- Spring Batch
- Arquitectura hexagonal
- Diseño Dirigido por el Dominio (DDD)
- Patrón SAGA
- Patrón Outbox
- Patrones de resiliencia

Con este trabajo, se busca explorar estas arquitecturas y patrones, las cuales han brindado soluciones a problemas comunes de complejidad, integración y escalabilidad. Inicialmente, se busca limitar el área de una entidad pública o entidad financiera, centrándolo a la notificación masiva de usuarios sobre sus multas y/o responsabilidades financieras en formato de un documento mediante computación distribuida, evitando tener un único bloque lógico. Buscando así contribuir al mejoramiento de la productividad y competitividad dentro del sector de la empresa que cuente con la necesidad de envíos masivos, para este alcance inicial no haremos uso de todos los patrones y arquitecturas mencionadas anteriormente.

II. MOTIVACIÓN

La motivación de esta propuesta surge de haber trabajado en un proyecto el cual contaba con una arquitectura monolítica, el cual tuvo un crecimiento acelerado en su volumen de datos a procesar, presentando problemas de escalabilidad e integración de nuevos servicios dado al acoplamiento del sistema. A raíz de esto al investigar, me encuentro que, aunque en el contexto colombiano, no se dispone de un informe en concreto de las organizaciones el cual detalle cuantas aún manejan o utilizan un sistema heredado o legacy, dado que mucha de esta información se mantiene de manera privada, podemos evidenciar que, dados algunos casos de estudio realizados en Colombia podrían indicar que aún existen empresas a nivel nacional operando con arquitecturas monolíticas, o incluso empresas que se encuentran en un camino de transición a arquitecturas más ajustables a los retos de la actualidad, que aunque se encuentren en esa transición pueden llegar a ser pospuestas por presentar una complejidad y costo.

Uno de estos ejemplos, está relacionado con la Universidad Nacional de Colombia, la cual detalla la evolución de la arquitectura de gestión de información de la universidad, sus desafíos, sus estrategias y la nueva arquitectura propuesta, para atender las nuevas necesidades de rendimiento, escalabilidad y mantenibilidad del sistema *[3],*presentando una propuesta de migración de arquitectura monolítica. De igual forma, la misma Escuela Colombiana de Ingeniería presenta un caso de migración de una arquitectura monolítica a microservicios en un sistema Fintech en Colombia *[4]*.

III. ESTADO DEL ARTE

Se tuvieron en cuenta diferentes criterios para la selección de los trabajos. Dentro de estos criterios se filtraron por documentos que hablaran de: arquitecturas para sistemas que trabajan grandes volúmenes de datos, herramientas tecnológicas empleadas para arquitecturas distribuidas, los desafíos que se presentan al generar una escalabilidad de servicios distribuidos, beneficios de las arquitecturas modernas frente a una monolítica, y soluciones con arquitecturas generadas en un contexto colombiano.

En la actualidad, el tema de estrategias para manejar grandes volúmenes de datos se encuentra muy avanzado, por lo que existen varias propuestas de investigación e implementaciones las cuales abordan problemáticas similares a la de promover el uso de arquitecturas distribuidas para el procesamiento de datos masivos. Entre estos, algunas incluso proponen la reingeniería y rediseño desde diferentes puntos de vista.

En cuanto a temas de los principales beneficios, retos, componentes, que puede llegar a tener una arquitectura distribuida como lo es la de microservicios *[5]* la ejemplifica basándose en contextos reales de empresas internacionales, tales como los son Amazon, eBay y Netflix. De las cuales destaca principalmente a Netflix, ya que esta tiene miles de clientes incluyendo televisores, teléfonos, consolas de videojuegos, tablets, entre otros, llegando a manejar miles de millones solicitudes por día. Esto es perfecto para analizar, dado el contexto que queremos orientar hacia el procesamiento masivo de notificaciones. Dentro de los principales aspectos para tener en cuenta, este nos menciona que este tipo de arquitecturas sufre con la complejidad que puede llegar a surgir al tratar de comunicar los servicios entre sí, incluyendo la importancia de manejar tolerancia a fallos, problemas que no ocurrían en una arquitectura donde todos los servicios se comunicaban dentro del mismo bloque lógico.

También me encontré con una propuesta de manejo de grandes volúmenes de datos para la gestión de prestaciones de salud *[6], e*l cual nos muestra un proyecto que pretende adaptar una arquitectura de microservicios para la migración de su arquitectura actual, lo que se destaca de este trabajo es el uso de tecnologías como Spring Framework , Spring Cloud, Spring gateway, Eureka Server ,Spring security, hystrix, herramientas tecnológicas válidas para el planteamiento de la arquitectura, herramientas que podemos validar y adaptar a nuestra solución, además esta está orientada en un contexto de grandes volúmenes de datos. Donde se basa en Spring Framework como plataforma para el desarrollo de su aplicación en Java orientada a construir lo servicios necesarios para ser reutilizables, al estar buscando arquitecturas distribuidas se apoya en Spring Cloud el cual es una extensión de Spring que contiene herramientas de comunicación de servicios por medio de patrones de arquitecturas distribuidas, por parte de Spring Api-Gateway es un componente de Spring Cloud el cual se encarga de distribuir las peticiones de los usuarios y los servicios entre los diferentes servicios, por parte del uso de Spring Security es un módulo que se le agrego a la arquitectura para tener seguridad, por ultimo hystrix es una herramienta para el tema de tolerancia a fallos al momento de tener comunicación entre servicios.

Para corroborar por qué una arquitectura como la de microservicios, las hace una de las más llamativas para el desarrollo de aplicaciones escalables, *[7]* nos comenta que esta arquitectura propone desarrollar servicios independientes, los cuales se deben de comunicar entre sí para suplir las necesidades de la lógica de negocio, manteniendo una descentralización del manejo de los datos.

Al tratarse de componentes independientes, este tipo de arquitectura permite agregar tantas instancias como sean necesarias para el procesamiento de solicitudes permitiendo así un escalamiento horizontal en el sistema, y no solo eso, esta arquitectura también permite que cada servicio sea dimensionado en recursos según la carga que este reciba, ocasionando que el rendimiento de los sistemas no se vea afectado, permitiendo a su vez que los servicios tenga una

escalabilidad vertical, no tan costosa como sería la de un monolito, como nos lo comenta *[9]*.

Por ultimo me encontré con el framework de codigo abierto el cual ofrece el mismo Spring, conocido como ***Spring Batch***, el cual se encuentra enfocado para la creación de sistemas o aplicaciones que cuenten con la necesidad de procesamiento de grandes volúmenes de datos, este permite hacer la lectura, procesamiento y manipulación de datos por medio de lotes, lo cual permite un mejor uso de memoria al no cargar todo en un solo golpe, y una mejor transaccionalidad en caso de interactuar con bases de datos, ya que cada lote se procesa con un único commit batch, este framework también presenta otras características como lo es el procesamiento en paralelo *[10]*.

IV. Arquitectura de solución

A. *Backend Facade (Gateway)*

- Una aplicación Spring Boot que actúa como la puerta de enlace API, desplegada en una instancia EC2 independiente.
- Este componente gestionará y dirigirá las solicitudes entrantes desde el cliente. Recibirá solicitudes POST del front-end, las validará y luego las enviará al servicio batch.

B. *Servicio Batch (Spring Batch)*

- Una aplicación para procesamiento batch implementada en Java y desplegada utilizando Docker. Manejará la lógica de leer, procesar y enviar los datos que se encuentren en DB para el servicio de notificación.

- El batch interactuará con una base de datos relacional para recuperar formas de contacto como correos a enviar.
- El batch procesara la data leída y la enviara la solicitud al servicio de notificaciones por medio del Gateway.

C. *Servicio Notificaciones*

- Servicio especializado a recibir peticiones para envió de correos, el cual se basará en el servicio SNS de AWS.

D. *Load Balancer (Spring Cloud Gateway)*

- Balanceador de carga que se encargara de distribuir las peticiones enviadas por el batch al servicio de notificaciones, usando el algoritmo Round Robin.

E. *Servicio Discovery (Spring Cloud Netflix)*

- Actúa como un servicio de registro de servicios donde cada instancia de un servicio se registra para que otros servicios puedan encontrarla y se puedan comunicar de manera distribuida.

F. *Servicio configuracion (Spring Cloud Config)*

- Servicio que almacena la configuración de los servicios, funcionando como un middleware entre los microservicios y el repositorio, para obtener la configuración.

Se propone la siguiente arquitectura:
como se muestra en la *Fig. 2*

V. Resultados

- Escalabilidad de servicio de notificaciones, según la necesidad de carga de procesamiento (vertical y horizontalmente).
- Configuración centralizada, la cual permite tener la configuración de varias instancias de forma consistente
- Manejo en la carga de trabajo de manera equitativa por medio de balanceadores algoritmos como lo es Round Robin.

VI. Conclusión

A. *Implicaciones para el futuro*

Este es un prototipo inicial para un sistema de envió masivo de notificaciones por medio del uso de servicios de AWS y herramientas proporcionadas por Spring Cloud, sin embargo, hay algunos temas e implicaciones que se deben de tener en cuenta para el futuro, para seguir enriqueciendo esta arquitectura:

- Dado los permisos con los que contamos en la cuenta de AWS, la aplicación será desplegada inicialmente en instancias de EC2, lo cual permite una escalabilidad sencilla dentro de la misma, sin embargo, para permitir una mejor escalabilidad de servicios ligeros, se debería investigar y configurar el despliegue en el servicio que ofrece Amazon de ECS *(Elastic Container Service)*.
- Como sabemos en este tipo de arquitecturas se presentan retos importantes, dentro de los cuales uno el cual se puede complementar en el futuro, es la tolerancia a fallos de la comunicación, se puede llegar a plantear patrones de tolerancia a fallos *[11]*, tales como:
  - Circuit Breaker
  - Rate Limiter
  - BulkHead
- Se plantea agregar una capa de seguridad en futuras integraciones, esto para garantizar la protección de los datos o incluso que solo

usuarios autorizados puedan consumir los servicios.

VII. FIGURAS

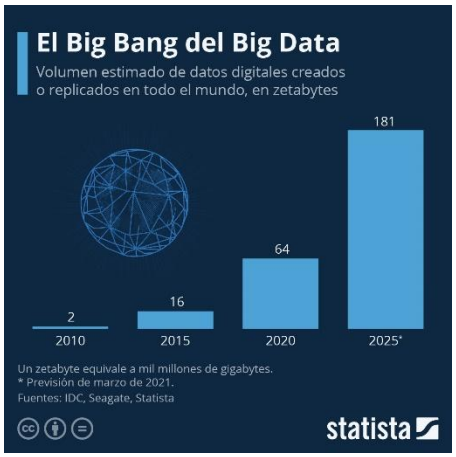

Fig. 1 El crecimiento del volumen de datos digitales creados y replicados a nivel mundial en zetabytes, statista

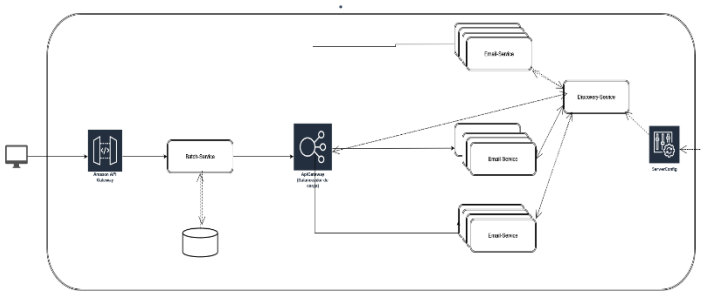

Fig. 2 Arquitectura AWS

VIII. BIBLIOGRAFIA

[1] Mónica Mena Roa, "El Big Bang del Big Data", Statista, 22 de octubre de 2021. Disponible: https://www.statista.com/statistics/1234567/big-bang-del-big-data.

[2] Avendaño, D. (2017c, Febrero 24). Multa de $ 840 millones a Bancolombia por fallas técnicas. El Tiempo. https://www.eltiempo.com/economia/empresas/multa-a-bancolombia-por-fallas-tecnicas-61496

[3] David, "Modelo de evolución arquitectónica de monolito a microservicios para el sistema de información ISys de la Dirección Nacional de Admisiones de la Universidad Nacional de Colombia," Unal.edu.co, 2022,Disponible: https://repositorio.unal.edu.co/handle/unal/81788.

[4] Nontoa Caballero, J. (2024). Migración de una arquitectura monolítica a microservicios: Un caso de aplicación en un sistema de Fintech en Colombia. Escuela Colombiana de Ingeniería, Disponible: https://repositorio.escuelaing.edu.co/handle/001/2876.

[5] RICHARDSON, Chris; SMITH, Floyd: Microservices - From Design to Deployment. In: Nginx

[6] RAMÍREZ CARVAJAL, Diego y SALAZAR ÁLVAREZ, Juan. Propuesta de modelo de arquitectura distribuida en microservicios para la gestión de prestaciones de salud en la empresa Conexia sede Bogotá. [en línea]. Bogotá : Universidad Cooperativa de Colombia, Facultad de Ingenierías, Ingeniería de Sistemas, Bogotá, 2020

[7] Lewis, J. and Fowler, M. (2014) Microservices, a Definition of This New Architectural Term.

[8] Lewis, J., & Fowler, M. (2014). *Microservices*. MartinFowler.com. Retrieved from https://martinfowler.com/articles/microservices.html

[9] M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," 2015 10th Computing Colombian Conference (10CCC), Bogota, Colombia, 2015, pp. 583-590, doi: 10.1109/ColumbianCC.2015.7333476.

[10] "Spring Batch," *Spring*, [En línea]. Disponible: https://spring.io/projects/spring-batch.

[11] "Resilience4j - Circuit Breaker", Baeldung. Disponible: https://www.baeldung.com/resilience4j.