UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN

SOFTWARE CON MENCIÓN EN DISEÑO DE ARQUITECTURA DE SISTEMAS

RPC-SO-34-NO.778-2021

OPCIÓN DE TITULACIÓN:

ARTÍCULOS PROFESIONALES DE ALTO NIVEL

TEMA:

IMPLEMENTACIÓN DE UNA ARQUITECTURA DISTRIBUIDA PARA ALMACENAR Y PROCESAR DATOS HIDRO-METEOROLÓGICOS EN TIEMPO REAL

AUTOR

JHONY CRISTIAN LLANO TUMBACO

DIRECTOR:

PAULINA ADRIANA MORILLO ALCÍVAR

QUITO – ECUADOR

2023

Autor(es):

Jhony Cristian Llano Tumbaco
Ingeniero en Software
Candidato a Magíster en Software con Mención en Diseño de Arquitectura de Sistemas por la Universidad Politécnica Salesiana – Sede Quito.
jllanot@est.ups.edu.ec

Dirigido por:

Paulina Adriana Morillo Alcívar.
Ingeniería en Electrónica y Telecomunicaciones por la Escuela Politécnica Nacional
Posgrado en Gestión de la Información por la Universidad Politécnica de Valencia
pmorillo@ups.edu.ec

Jhony Cristian Llano Tumbaco
IMPLEMENTACIÓN DE UNA ARQUITECTURA DISTRIBUIDA PARA ALMACENAR Y PROCESAR DATOS HIDRO-METEOROLÓGICOS EN TIEMPO REAL

Implementación de una arquitectura distribuida para almacenar y procesar datos hidro-meteorológicos en tiempo real.

Jhony Cristian Llano Tumbaco

Universidad Politécnica Salesiana - Ecuador

Resumen

En el ámbito de la hidro-meteorología, la adquisición y análisis de datos en tiempo real son fundamentales para diversas aplicaciones críticas. Sin embargo, el creciente volumen de los datos genera desafíos en términos de almacenamiento y procesamiento. Ergo, este trabajo presenta la solución de una arquitectura de bases de datos distribuida destinada a la gestión de grandes conjuntos de datos hidrometeorológicos del Instituto Nacional de Meteorología e Hidrología del Ecuador.

Para resolver esta problemática, se analizó las condiciones de arquitectura actual y se implementó una arquitectura distribuida por medio de PostgreSQL y Citus. Este enfoque permitió la distribución y manejo eficiente de los datos, facilitando su almacenamiento, recuperación y análisis en tiempo real, a través de múltiples nodos de procesamiento.

Los resultados de la implementación demostraron una notable mejora en términos de escalabilidad, velocidad y confiabilidad del sistema de almacenamiento y procesamiento de datos. Se destaca especialmente la capacidad de manejar con eficacia picos de demanda y grandes volúmenes de datos, garantizando tiempos de respuesta rápidos.

Esta innovación en la gestión de datos hidrometeorológicos resalta la importancia de adoptar arquitecturas distribuidas y tecnologías de procesamiento en tiempo real, estableciendo un precedente relevante para futuras investigaciones y aplicaciones en el campo.

Abstract

In the field of hydrometeorology, the acquisition and analysis of real-time data are fundamental for various critical applications. However, the increasing volume of data poses challenges in terms of storage and processing. Ergo, this work presents the solution of a distributed database architecture aimed at managing large sets of hydrometeorological data from the National Institute of Meteorology and Hydrology of Ecuador.

To address this issue, the current architecture conditions were analyzed and a distributed architecture was implemented using PostgreSQL and Citus. This approach allowed for the efficient distribution and handling of data, facilitating its storage, retrieval, and real-time analysis, through multiple processing nodes.

The implementation results showed a notable improvement in terms of scalability, speed, and reliability of the data storage and processing system. It especially highlights the ability to effectively handle demand peaks and large volumes of data, ensuring quick response times.

This innovation in the management of hydrometeorological data emphasizes the importance of adopting distributed architectures and real-time processing technologies, setting a significant precedent for future research and applications in the field.

Palabras clave— Citus; Postgres; Computación en la Nube; Escalabilidad; Tolerancia a Fallos; Integridad de Datos; Replicabilidad de Datos .

1 Introducción

La obtención de información meteorológica es crucial para entender y prever las tendencias climáticas, así como en la adopción de decisiones vinculadas con la gestión de desastres naturales, agricultura, infraestructura, etc. Sin embargo, a medida que se recopila la información meteorológica a través de una amplia variedad de instrumentos y sensores, surge un desafío igualmente importante relacionado con el almacenamiento de estos datos. La enorme cantidad de información climática generada a diario, que incluye mediciones de temperatura, humedad, viento, precipitación y otros parámetros, plantea desafíos significativos en términos de capacidad de almacenamiento, organización y accesibilidad, lo que requiere soluciones tecnológicas innovadoras y estrategias de gestión de datos eficientes.

En esta situación, los sistemas de almacenamiento de información descentralizadas emergen como una opción sobresaliente para el almacenamiento y gestión de datos climáticos, gracias a su habilidad para administrar vastas sumas de información y escalar eficientemente Schletz et al. (2022).

En la actualidad, las bases de datos descentralizadas han adquirido una gran relevancia en el funcionamiento de las organizaciones globales. Debido a que se requiere que los sistemas de información ofrezcan confiabilidad, escalabilidad y acceso rápido. Del mismo modo, los datos han evolucionado en diversas formas y soportes. Por lo tanto, para dar respuesta a estas necesidades, así como los complejos los complejos requerimientos de red y el incremento en el volumen de datos, han surgido los sistemas distribuidos de almacenamiento de datos. Un Sistema distribuido de Almacenamiento de Datos también conocido como DDBS por sus siglas en inglés (*Distributed Data Storage System*), es una infraestructura de almacenamiento de datos que se caracteriza por la dispersión de los datos en múltiples ubicaciones físicas o servidores interconectados en una red informática. Un DDBS permite que los datos se almacenen de manera descentralizada en varios nodos o servidores, lo que mejora la escalabilidad y la tolerancia a fallos. Los datos se dividen y replican entre estos nodos Zhang et al. (2021), lo que contribuye a la redundancia y a mantener la integridad de los datos en caso de inconvenientes técnicos. Además, estos sistemas pueden mantener una relación lógica entre los datos a pesar de su ubicación dispersa, lo que facilita su gestión y acceso Zheng et al. (2022).

Existen numerosos campos de aplicación en la industria donde se procesan y recuperan datos de manera distribuida como la utilización de bases de datos distribuidas para portales de viajes en línea que operando en ambientes *cloud* Barua et al. (2023). Este estudio explora cómo las bases de datos distribuidas, junto con las aplicaciones de microservicios, pueden prevenir fallas completas del sistema, manejar grandes volúmenes de datos y balancear la carga de una aplicación, contribuyendo así a la eficiencia y resiliencia de los portales de viajes en línea. Aunque no se proporcionan detalles adicionales en los resultados de búsqueda, esta investigación resalta la relevancia de las bases de datos distribuidas en la mejora de la operatividad y eficiencia en entornos de alta demanda y datos masivos.

Asimismo, un número creciente de empresas e instituciones están requiriendo incorporar soluciones distribuidas para el almacenamiento y gestión de sus datos. Es el caso del Instituto Nacional de Meteorología e Hidrología (INAMHI), que tradicionalmente, ha mantenido una base de datos relacional centralizada, esta base de datos tiene un control y gestión de datos estructurado, pero que ha empezado a mostrar signos de estrés ante el creciente volumen de datos y las demandas de acceso en tiempo real. Los problemas de sobrecarga y la latencia en el acceso de los datos se convirtieron en preocupaciones palpables. Además, la recuperación de datos en caso de fallos y la continuidad del servicio son cuestiones que exigían una revisión de la infraestructura existente.

El INAMHI es una institución especializada en meteorología e hidrología cuya misión es recolectar, procesar y difundir información climática en todo el Ecuador. Gracias a su extensa red de estaciones meteorológicas distribuidas a lo largo del territorio ecuatoriano, el INAMHI puede proporcionar datos precisos y actualizados acerca de las condiciones climáticas en el país. Esta información incluye métricas como la temperatura, precipitación, índice de humedad, velocidad, dirección del viento, entre otros

aspectos climáticos INAMHI (2023). Por lo tanto, dada la importancia y el volumen de información que maneja el INAMHI. Esta institución ha considerado mejorar su infraestructura y migrar la DB actual a una DDBS. La implementación de la arquitectura DDBS en el almacenamiento y gestión de datos climáticos brindará al sistema una mayor capacidad de escalabilidad y disponibilidad de la información, así como mejoras en seguridad y replicación de datos Nidzwetzki and Güting (2017) Mazumdar et al. (2019). Además, su uso se ha vuelto cada vez más popular en la investigación científica para analizar y modelar datos climáticos, lo que contribuye significativamente a una comprensión y predicción más precisas del clima. No obstante, resulta fundamental considerar los desafíos asociados con la administración y gestión del sistema, así como la sincronización de datos entre los nodos, a fin de asegurar un funcionamiento óptimo del mismo.

Ergo, este trabajo propone implementar una arquitectura distribuida para abordar aspectos como la carga y procesamiento de datos climáticos del INHAMI. A través de la implementación de Citus, una extensión de PostgreSQL diseñada para gestionar bases de datos distribuidas Cubukcu et al. (2021), se busca optimizar la gestión de grandes volúmenes de datos climáticos, reduciendo los cuellos de botella, disminuyendo los tiempos de respuesta y aprovechando al máximo la capacidad de I/O y computación disponible en un clúster de servidores.

De esta forma, se ha estructurado el documento en cinco secciones. En la sección 2 se detalla el proceso realizado para evaluar la base de datos actual y diseñar la nueva arquitectura. En la sección 3 se muestra el desarrollo de la implementación. En la sección 4 se indican los resultados de la evaluación. Finalmente, en la sección 5, se presentan las conclusiones y las recomendaciones de este trabajo.

2 Metodología

La Figura 1 presenta un esquema detallado del proceso de migración de la base de datos relacional actual del INHAMI a una base de datos relacional distribuida. Este proceso, presenta cuatro fases. La etapa inicial se centra en el análisis y diagnóstico de la arquitectura actual para conocer su estructura, sus funcionalidades y los tipos de datos que almacena. También se identifican las tablas clave, se determina la estrategia de distribución, se evalúa la viabilidad de la migración y se selecciona la tecnología empleada para la nueva arquitectura.

Posteriormente, se realiza el diseño de la arquitectura distribuida, la configuración de parámetros y ajustes preliminares, el respaldo de la DB para asegurar la integridad y disponibilidad de los datos durante la transición, y finalmente, el proceso de desarrollo se realiza la migración per se, que implica la transición efectiva de datos y aplicaciones. La última fase del proceso consiste en la evaluación de la arquitectura propuesta, realizando una comparativa entre la arquitectura distribuida y la arquitectura actual. El objetivo de la evaluación es medir y validar la eficacia, rendimiento y beneficios de la migración de la arquitectura.

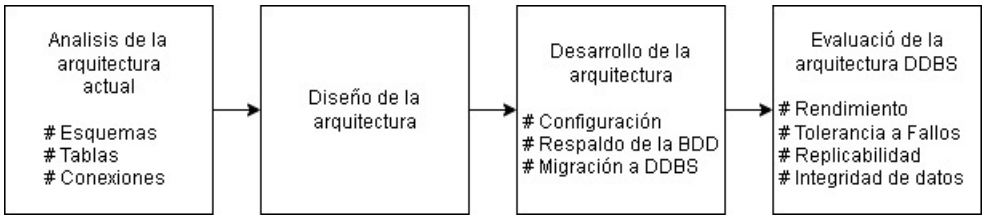

Figura 1: Proceso de migración de una DB relacional centralizada a una distribuida.

La transición de un sistema de almacenamiento de información relacional centralizada a un sistema de almacenamiento de datos distribuida puede ser un proceso sumamente complejo. Así la metodología de *sharding* emerge como una valiosa técnica que fragmenta la base de datos en múltiples partes, conocidos como *shards*, y los distribuye entre diversos nodos dentro de un clúster lo que permite el procesamiento paralelo y la asignación más equitativa de la carga de trabajo. También, resulta en un mejor rendimiento y una mayor escalabilidad en comparación con las bases de datos relacionales centralizadas. Además, el *sharding* ofrece ventajas significativas en términos de tolerancia a fallos y disponibilidad, ya que la pérdida de un nodo o *shard* no afecta la disponibilidad de los demás da Silva and Lima (2023). Por lo

tanto, esta metodología resulta de gran utilidad al considerar la migración del sistema de almacenamiento de información del INAHMI hacia una base de datos distribuida relacional. En este caso se propone un sistema de gestión de base de datos relacional configurado en Postgres y Citus. Esta configuración se basa en tres nodos seleccionados y configurados para lograr un equilibrio entre rendimiento y disponibilidad. Cada uno de estos nodos opera con una máquina robusta, con capacidad de procesamiento y memoria RAM adecuadas para manejar cargas de trabajo intensivas Bakli et al. (2019). Además, se establece una conexión de red confiable y de alta velocidad entre los nodos para asegurar una comunicación eficaz. También es necesario la implementación de un mecanismo de respaldo y recuperación para proteger los datos en caso de fallos. En conjunto, estas características permiten aprovechar las capacidades de escalabilidad y rendimiento que Citus ofrece en la configuración de tres nodos (Figura 2), brindando al INAHMI una infraestructura de base de datos altamente eficiente y confiable.

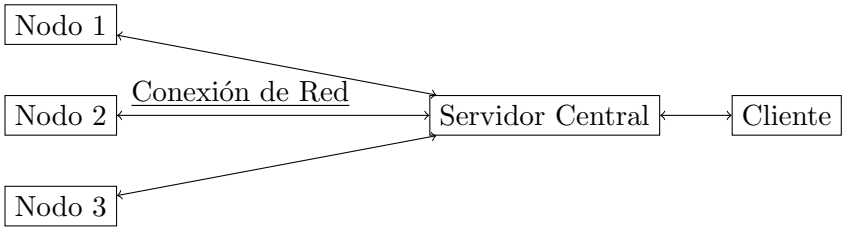

Figura 2: Características de Nodos y Servidor en una Configuración de PostgreSQL con Citus para INAHMI.

3 Implementación de la arquitectura distribuida

3.1 Análisis de la base de datos actual

El diseño de la arquitectura actual del INAMHI almacena, gestiona y organiza los datos climáticos que se recopilan en las estaciones meteorológicas distribuidas en todo el país. La base de datos está compuesta por diversas tablas relacionadas entre sí, y su estructura puede variar según los datos que se estén almacenando.

La estructura de la DB del INAMHI se refleja en la Figura 3, la cual se organiza en sus esquemas y tablas, facilitando la gestión de un volumen significativo de información meteorológica. Esta compleja base de datos se divide en cuatro esquemas principales que atienden a diferentes aspectos de la información meteorológica: el esquema administrativo, el esquema automático, el esquema convencional y el esquema seguridad.

El esquema administrativo es el núcleo central de la base de datos, incluye 40 tablas que abarcan detalles esenciales sobre las estaciones meteorológicas automáticas y convencionales. Este esquema proporciona una base sobre la cual operan otros esquemas, ofreciendo información vital como unidades de medida, variables meteorológicas, y ubicaciones geográficas, fundamentales para la precisión en la interpretación de los datos meteorológicos.

Del mismo modo, el esquema automático y el esquema convencional están diseñados específicamente para manejar la información generada por diferentes tipos de estaciones meteorológicas. El esquema automático contiene 136 tablas y se encarga de registrar y validar las mediciones obtenidas de las estaciones automáticas. Esto es crucial para mantener la precisión, ya que estas estaciones suelen generar grandes volúmenes de datos que requieren un procesamiento cuidadoso y validación constante. En contraste, el esquema convencional incluye 42 tablas y se ocupa de las mediciones de las estaciones convencionales, asegurando que los datos recopilados manualmente sean precisos y consistentes. El esquema seguridad contiene la información de los usuarios que utilizan los sistemas.

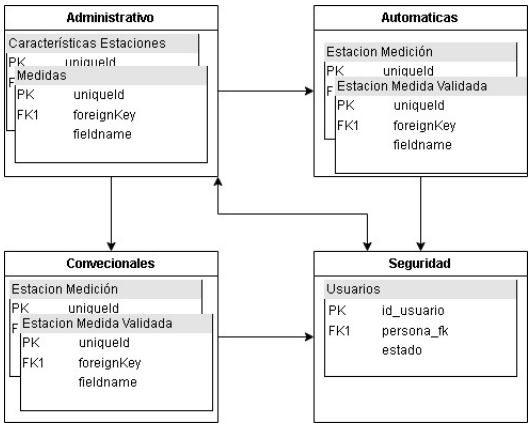

Figura 3: Diagrama de Relaciones entre los esquemas: Administrativo, Automáticas, Convencionales y Seguridad.

En cuanto a las tablas específicas dentro de estos esquemas, cada una sirve para un propósito único y esencial. La tabla Estaciones, por ejemplo, almacena información detallada sobre cada estación meteorológica, incluyendo su ubicación, elevación y los instrumentos utilizados. Esta información sirve para contextualizar los datos recopilados, permitiendo a los investigadores y científicos ajustar cualquier comparación de datos por diferencias en la geografía o el equipo utilizado.

La tabla de Mediciones registra todos los datos climáticos individuales. Cada entrada en esta tabla incluye detalles como la temperatura, la humedad, las precipitaciones y la velocidad del viento, así como una marca de tiempo, proporcionando un registro histórico completo, que es indispensable para el análisis climático y la predicción meteorológica.

Además, la base de datos mantiene la coherencia y el orden mediante el uso de la tabla de Variables y la tabla de Unidades. La Tabla de Variables describe qué variables climáticas se están midiendo. Paralelamente, la tabla Unidades especifica las unidades de medida para estas variables, garantizando la uniformidad en la comparación de datos a través de diferentes sistemas de medición.

En conjunto, la estructura detallada de esta base de datos está diseñada para preservar la integridad de la información meteorológica y facilitar su análisis y comprensión. Los esquemas y tablas interactúan de manera que permiten una recopilación de datos precisa, un almacenamiento organizado y un acceso eficiente, sirviendo como una infraestructura robusta para el manejo de la información meteorológica crítica.

Por otro lado, dado que el personal del INAMHI está familiarizado con PostgreSQL. Se propone una arquitectura de base de datos relacional basado en Citus. Citus es una extensión de PostgreSQL que facilita el escalado horizontal de datos, este proceso es crucial para manejar los crecientes volúmenes de datos climáticos. Existen varios casos de uso de Citus en empresas como Microsoft Hahn et al. (2021), Heap y Sentry demuestran su eficacia en el manejo de *petabytes* de datos y en la mejora del rendimiento de las operaciones de base de datos. La transición a Citus puede ser menos compleja, dado que extiende las funcionalidades de PostgreSQL, minimizando así los costos y tiempos asociados con la migración. La distribución de consultas y operaciones entre múltiples nodos que ofrece Citus es fundamental para el procesamiento eficiente de datos climáticos en tiempo real o cercano al real, facilitando el análisis oportuno para la toma de decisiones Cubukcu et al. (2021). Aunque se requiere una evaluación detallada de los costos y la configuración, la inversión inicial podría justificarse por los beneficios en escalabilidad y rendimiento. Además, Citus cuenta con una comunidad activa y soporte, lo que añade confianza para resolver posibles problemas durante la implementación y mantenimiento de esta solución tecnológica.

3.2 Diseño de la arquitectura de la DDBS

La utilización de Citus en la arquitectura de la DDBS ayuda a dividir las consultas grandes en tareas más manejables, conocidas como consultas fragmentadas. Esto es beneficioso porque permite que diferentes partes de una consulta se procesen simultáneamente en paralelo Hahn et al. (2021), lo que a su vez ayuda a evitar la sobrecarga que podría ocurrir debido a la latencia de la red o al consumo de recursos en los

nodos que están procesando las consultas.

Citus descompone cada consulta entrante en tareas específicas para cada fragmento Vaisman and Zimányi (2022). Estas tareas son encoladas y ejecutadas cuando hay conexiones disponibles hacia los nodos trabajadores correspondientes. Este proceso garantiza una gestión eficiente de las conexiones y maximiza el rendimiento de la consulta. Al diseñar la arquitectura de Citus, es fundamental comprender esta gestión de conexiones para asegurar una operación eficiente y que se adapte a las demandas específicas de la carga de trabajo.

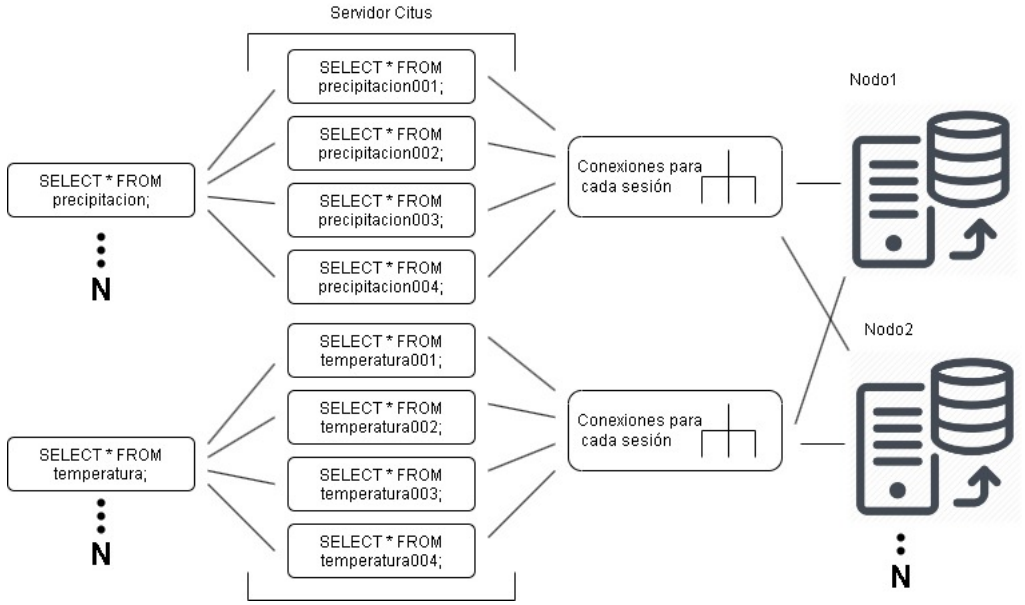

Figura 4: Distribución de Consultas y Conexiones en Arquitectura Citus con Nodos de Datos.

La Figura 4 presenta la estructura de la DDBS. En el lado izquierdo, hay dos consultas principales: "SELECT * FROM precipitacion;" y "SELECT * FROM temperatura;". Estas consultas se dirigen hacia un "Servidor Citus" que se visualiza en el centro de la imagen.

El Servidor Citus actúa como un coordinador que descompone estas consultas generales en consultas más específicas. Por ejemplo, la consulta de precipitación se divide en *"SELECT * FROM precipitacion001;"*, *"SELECT * FROM precipitacion002;"*, y así sucesivamente. De manera similar, la consulta de temperatura se descompone en consultas específicas para "temperatura001", "temperatura002", etc.

Una vez que el Servidor Citus ha descompuesto las consultas, estas se envían a través de "Conexiones para cada sesión", representadas por rectángulos con líneas. Estas conexiones son canales individuales a través de los cuales se ejecutan las consultas descompuestas.

En el extremo derecho de la Figura 4, hay n nodos, etiquetados como "Nodo1", "Nodo2", etc. Estos nodos son los destinos finales de las consultas descompuestas y son los lugares desde donde se recuperan los datos. Las flechas entre las conexiones de sesión y los nodos indican el flujo de las consultas y la recuperación de datos.

3.3 Desarrollo de la arquitectura de la DDBS

La implementación de la arquitectura en el INAHMI tiene las siguientes características: los nodos están equipados con procesadores Intel Xeon con 8 núcleos y 64 GB de RAM, respaldados por unidades SSD de 1 TB. Tiene una conectividad de red Gigabit Ethernet con sistema operativo basados en Linux con Ubuntu server v22.04. En cuanto al servidor central, cuenta con un procesador de 16 núcleos y 64 GB de RAM, almacenamiento SSD 2T, conectividad de red de 10 Gigabit Ethernet, un sistema de respaldo externo y herramientas de monitoreo como Prometheus y Grafana para supervisar el rendimiento.

La documentación sobre el protocolo que se realizó para la gestión y migración de la DB a la arquitectura planteada está disponible en el siguiente enlace: https://github.com/crisdan1991/config-citus-c.

El protocolo empieza con la configuración del Host, donde se introducen los comandos necesarios para instalar la herramienta y configurar nodos relacionados, estableciendo así un ambiente propicio para las operaciones subsecuentes de la DDBS.

La sección dedicada a la exportación de datos indica cómo llevar a cabo la exportación tanto de los datos como de la estructura de la DB actual utilizando el programa pg_dump.exe. Se presenta un comando específico para la exportación exclusiva de los datos y otro distinto para la estructura de la base de datos. Ambos comandos cuentan con opciones específicas que se adaptan según las necesidades del proceso de exportación. Esta herramienta nos permite asegurar un respaldo actualizado de la base de datos y facilita la migración de su estructura o la creación de un esquema de distribución.

En la siguiente fase, titulada Distribución de Tablas y Creación de Llaves Foráneas, es vital identificar los tipos de tablas presentes. Una vez que se ha identificado la clave de distribución, se debe examinar el esquema para determinar el manejo de cada tabla y las posibles modificaciones necesarias.

Al seleccionar una columna de distribución, es esencial elegir una con alta cardinalidad, evitando columnas con valores limitados, como un campo de "estado" que solo tiene unas pocas opciones, ya que esto restringe el número de fragmentos y nodos que pueden procesar los datos. Es recomendable que estas columnas de alta cardinalidad se utilicen frecuentemente en cláusulas de agrupación o como claves de unión. Como ejemplo, en los esquemas Convencionales y Automáticos, cada uno posee dos tablas relacionadas con un tipo de variable, ya sea Nubosidad o Temperatura, y son precisamente estas, las que se deben distribuir.

Es crucial elegir una columna con una distribución uniforme para evitar el desequilibrio en la carga de trabajo entre nodos. Al distribuir tablas de hechos y dimensiones, es importante hacerlo en sus columnas comunes y considerar la frecuencia y tamaño de las uniones para co-ubicación. Si una tabla de dimensiones no puede ser co-ubicada con la tabla de hechos, distribuir copias de esta a todos los nodos como una tabla de referencia, puede mejorar el rendimiento de las consultas como el esquema Administrativo y de Seguridad. Teniendo en cuenta estos aspectos, se segmenta el esquema de la base de datos vigente en tres archivos: base de datos no relacionada, distribución de las tablas y establecimiento de relaciones.

La sección de Preparación del Ambiente para migración, detalla los pasos para configurar el ambiente para una migración efectiva, incluyendo la creación de la DB en el servidor y los nodos, la configuración de nodos y la adición de nodos al servidor, estableciendo así las bases para una migración de datos exitosa.

Finalmente, la sección de Carga de Base de Datos y Migración de Datos proporciona una serie de comandos que facilitan la carga, la distribución de esquemas de tablas, la carga de llaves foráneas y la migración de los datos, culminando así el protocolo técnico para la gestión y migración efectiva de bases de datos en PostgreSQL.

4 Evaluación de la arquitectura distribuida

En la infraestructura proporcionada, se cuenta con dos servidores principales, ambos equipados con características idénticas. En uno de estos servidores se encuentra la base de datos centralizada del INAMHI, donde todos los datos están almacenados en un solo lugar. El otro servidor se utiliza para implementar una DDBS, aprovechando la capacidad de distribuir la carga de trabajo entre múltiples nodos como se detalló el la sección 3.3. Esta configuración permite comparar y contrastar el rendimiento entre una base de datos centralizada y una distribuida, aprovechando la capacidad de escalabilidad y paralelismo ofrecida por los nodos adicionales.

El rendimiento de la base de datos se evaluó utilizando varios *scripts* que representan distintas operaciones asociadas a la administración de una base de datos en PostgreSQL y Citus. Uno de los *script* ejecuta una consulta SELECT con el propósito de recuperar información, el otro *script* se encarga de insertar datos generados de manera aleatoria, tal como se ilustra en la Figura 4.

Ambos *scripts* están diseñados para medir el tiempo que tardan en ejecutarse, tanto a nivel de hilo individual como a nivel global, el segundo *script* genera e inserta datos aleatorios en una tabla específica, también en 10 hilos, cada uno realiza 1000 inserciones. Los datos generados incluyen valores de fecha y hora, así como valores numéricos aleatorios para representar minutos, y se registran junto con identificadores fijos para estación y usuario. La inserción de datos se realiza mediante una consulta SQL de inserción. Ambos scripts utilizan la biblioteca *threading* para crear múltiples hilos y la biblioteca

psycopg2 para interactuar con la base de datos, y registran el tiempo total que tarda la ejecución en cada hilo y a nivel global, proporcionando una medida de la duración de estas operaciones.

Para evaluar la arquitectura se desarrolló un *script* en Python diseñado para insertar y buscar datos de manera concurrente utilizando múltiples hilos. Este *script* se ejecutó sobre la arquitectura propuesta y la actual. La conexión con DB se realizó con la librería psycopg2 y la generación de datos ficticios con la librería Faker. Al finalizar, el programa imprime la duración total de la ejecución y la duración de cada hilo en particular como se presenta en la Figura5.

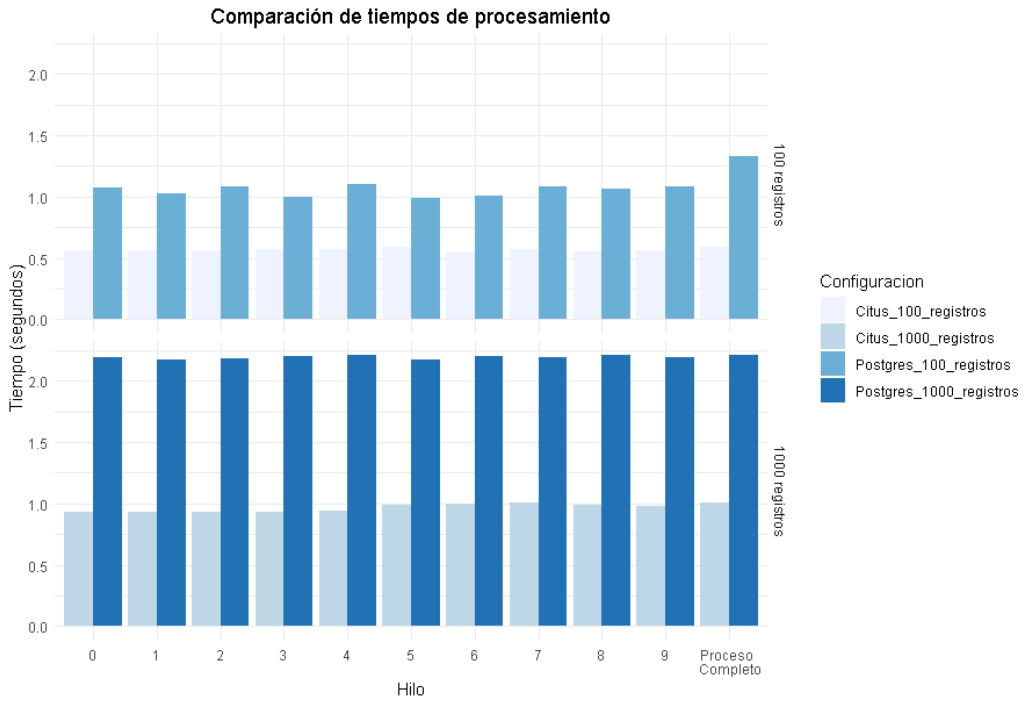

Figura 5: Comparación de tiempos de procesamiento entre la configuración solo con Postgres y la configuración distribuida con Citus y Postgres.

El análisis de los tiempos registrados en las operaciones realizadas en las bases de datos refleja una mejora significativa en la eficiencia al migrar de una DB solo con Postgres a una DDBS utilizando Citus junto con Postgres. A continuación, se detallan los puntos destacados del análisis:

La Figura 5, compara el rendimiento de dos configuraciones diferentes en términos de tiempo de procesamiento: una utilizando solo Postgres y otra distribuida con Citus y Postgres. El rendimiento se evaluó en función del tiempo que toma procesar 100 y 1000 registros para diez hilos distintos.

Al analizar la configuración que utiliza solo Postgres, se observa que el tiempo promedio de procesamiento para 100 registros oscila entre 0.99 y 1.32 segundos, con un promedio aproximado de 1.08 segundos. Para 1000 registros, el tiempo varía levemente entre 2.17 y 2.21 segundos, promediando alrededor de 2.19 segundos. Es destacable que el tiempo de procesamiento muestra una variación mínima entre los distintos hilos de ejecución para 1000 registros.

Por otro lado, en la configuración distribuida con Citus y Postgres, el tiempo promedio para procesar 100 registros se sitúa entre 0.54 y 0.59 segundos, con un promedio cercano a 0.57 segundos. Para 1000 registros, los intervalos de tiempo varían entre 0.93 y 1.01 segundos, con una media aproximada de 0.97 segundos.

Al comparar ambas configuraciones, es evidente que la distribuida con Citus y Postgres tiene un rendimiento superior, siendo aproximadamente el doble de rápida en el procesamiento de registros que la configuración solo con Postgres. En términos específicos, se observa una mejora del 47 % en el procesamiento de 100 registros y del 56 % para 1000 registros con la configuración distribuida.

Se observa una mayor consistencia en los tiempos de completado de los hilos en la configuración con Citus y Postgres en comparación con la configuración solo con Postgres. Esta consistencia podría

traducirse en una predictibilidad mejorada en el desempeño de la DB.

Esto reitera el beneficio de utilizar una base de datos distribuida para manejar operaciones en un entorno que podría estar experimentando un alto volumen de consultas o transacciones.

Tipo de Prueba	**Descripción**	**Procedimiento**	**Resultados DB actual**	**Resultados DDBS propuesta**
Tolerancia a Fallos	Verificar que el sistema continúa funcionando si un nodo falla.	1. Desconectar un nodo. 2. Realizar consultas. 3. Observar el comportamiento.	El sistema deja de funcionar	El sistema sigue funcionando correctamente
Replicabilidad	Asegurarse de que los datos estén replicados correctamente en todos los nodos.	1. Insertar datos en un nodo. 2. Verificar otros nodos.	No existe replicabilidad	Los datos se encuentra en todos los nodos
Integridad de Datos	Confirmar que los datos escritos son los mismos que los leídos.	1. Escribir datos. 2. Leer datos. 3. Comparar.	Los datos recuperados son exactamente como los ingresados	Los datos recuperados son exactamente como los ingresados

Tabla 1: Tabla de pruebas para las arquitecturas.

Mientras que una configuración estándar de PostgreSQL puede ser susceptible a puntos de fallo únicos, afectando la disponibilidad general del sistema en caso de interrupciones, la arquitectura de Citus ha mostrado una resiliencia superior Felius (2022). Durante las pruebas realizadas en la Tabla 1 de desconexión de nodos, donde una base de datos PostgreSQL típica habría experimentado tiempos de inactividad significativos o incluso una parada total, DDBS propuesta redistribuyó las cargas y procesó las consultas sin interrupción notable. Esta diferencia crucial subraya la robustez de Citus en mantener la continuidad del negocio, reduciendo efectivamente los riesgos asociados con la duración de la inactividad y la pérdida de acceso a datos cruciales.

En un entorno PostgreSQL, la integridad de los datos puede comprometerse bajo condiciones de alta concurrencia o cuando se presentan fallos inesperados, ya que las transacciones podrían no completarse correctamente. En contraste, Citus maneja estas situaciones con una precisión excepcional. Durante las pruebas, Citus no solo mantuvo la integridad de los datos bajo cargas pesadas, sino que también previno la pérdida de datos en casos de fallos. La capacidad de Citus para realizar automáticamente *rollbacks* seguros o completar transacciones a pesar de las interrupciones, asegura que cada bit de información se mantenga íntegro y exacto.

En la arquitectura actual del INAMHI, la replicación puede requerir configuraciones manuales complejas y no se puede garantizar siempre la coherencia inmediata entre los nodos. Esta limitación fue notablemente superada por la arquitectura propuesta en las pruebas. Al realizar modificaciones en la DB, se observa que, donde la arquitectura actual podría haber requerido procedimientos adicionales y tiempo de latencia para reflejar los cambios en todos los nodos, Citus aseguró una replicación casi instantánea y coherente. Esta eficiencia elimina las ventanas de inconsistencia que podrían afectar las operaciones del negocio, ofreciendo una confianza superior en la homogeneidad de los datos consultados en diferentes puntos del sistema.

5 Conclusiones y Recomendaciones

Este trabajo presenta la implementación de la migración de la DB centralizada del INAMHI a una arquitectura de DDBS, por medio de Citus y Postgres. Como muestran los resultados de la evaluación, la nueva arquitectura presenta una respuesta casi el doble de rápida que la arquitectura centralizada, lo que aumenta las posibilidades de su uso en aplicaciones en tiempo real y de alta confiabilidad.

Los tiempos de procesamiento de la nueva arquitectura se han reducido casi a la mitad respecto a los tiempos de la DB centralizada, lo que indica una mayor eficiencia en la gestión de las consultas, mayor consistencia en los tiempos de procesamiento entre los diferentes hilos y mejor rendimiento de la DB.

La tolerancia a fallos es mayor debido a que si un nodo se cae o se encuentra fuera de servicio los datos se encuentran replicados en otros nodos y se encuentran disponible en tiempo real, permitiendo la continuidad de las operaciones sin interrupciones significativas.

Por otro lado, la replicabilidad se logra mediante la implementación de los nodos. Cada dato ingresado en la base principal es replicado automáticamente en nodos secundarios, asegurando no solo una mayor disponibilidad de la información, sino también redundancia en caso de fallos. Este sistema de replicación opera en tiempo real y cuenta con protocolos, asegurando que todas las copias de los datos permanezcan sincronizadas entre los nodos.

Los resultados de la Figura 5, muestran también que la distribución de la carga en diferentes nodos es equilibrada, lo que permite un procesamiento eficiente y equitativo de las consulta, especialmente en entornos con un alto volumen de consultas o transacciones, como en el caso de datos climáticos.

Por otro lado, se recomienda establecer un sistema de monitoreo continuo para evaluar el rendimiento de la DB en diferentes condiciones de carga y optimizar la configuración según sea necesario. De igual forma, es recomendable que el equipo técnico a cargo del sistema, cuente con la capacitación adecuada en la gestión y mantenimiento de una base de datos distribuida.

Referencias

M. Bakli, M. Sakr, and E. Zimanyi. Distributed moving object data management in mobilitydb. In *Proceedings of the 8th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369664. doi: 10.1145/3356999.3365467. URL https://doi.org/10.1145/3356999.3365467.

B. Barua, M. Whaiduzzaman, M. Mesbahuddin Sarker, M. Shamim Kaiser, and A. Barros. Designing and implementing a distributed database for microservices cloud-based online travel portal. In S. Shakya, K.-L. Du, and K. Ntalianis, editors, *Sentiment Analysis and Deep Learning*, pages 295–314, Singapore, 2023. Springer Nature Singapore. ISBN 978-981-19-5443-6.

U. Cubukcu, O. Erdogan, S. Pathak, S. Sannakkayala, and M. Slot. Citus: Distributed postgresql for data-intensive applications. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, page 2490–2502, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383431. doi: 10.1145/3448016.3457551. URL https://bibliotecas.ups.edu.ec:2582/10.1145/3448016.3457551.

L. F. da Silva and J. V. F. Lima. An evaluation of relational and nosql distributed databases on a low-power cluster. *The Journal of Supercomputing*, 79(12):13402–13420, 2023. ISSN 1573-0484. doi: 10.1007/s11227-023-05166-7. URL https://doi.org/10.1007/s11227-023-05166-7.

C. Felius. *Assessing the performance of distributed PostgreSQL*. PhD thesis, Universiteit van Amsterdam, 2022.

S. M. L. Hahn, I. Chereja, and O. Matei. Comparison of the performance of newsql databases based on linux os. In R. Silhavy, P. Silhavy, and Z. Prokopova, editors, *Data Science and Intelligent Systems*, pages 372–384, Cham, 2021. Springer International Publishing. ISBN 978-3-030-90321-3.

INAMHI. Instituto nacional de meteorología e hidrología, 2023. URL https://inamhi.gob.ec/.

S. Mazumdar, D. Seybold, K. Kritikos, and Y. Verginadis. A survey on data storage and placement methodologies for cloud-big data ecosystem. *Journal of Big Data*, 6:1–37, 12 2019. ISSN 21961115. doi: 10.1186/S40537-019-0178-3/TABLES/4. URL https://bibliotecas.ups.edu.ec:3401/articles/10.1186/s40537-019-0178-3https://bibliotecas.ups.edu.ec:3401/article/10.1186/s40537-019-0178-3.

J. K. Nidzwetzki and R. H. Güting. Distributed secondo: an extensible and scalable database management system. *Distributed and Parallel Databases*, 35:197–248, 12 2017. ISSN 15737578. doi: 10.1007/S10619-017-7198-9/FIGURES/30. URL https://bibliotecas.ups.edu.ec:3401/article/10.1007/s10619-017-7198-9.

M. Schletz, A. Hsu, B. Mapes, and M. Wainstein. Nested climate accounting for our atmospheric commons—digital technologies for trusted interoperability across fragmented systems. *Frontiers in Blockchain*, 4, 2022. ISSN 2624-7852. doi: 10.3389/fbloc.2021.789953. URL https://www.frontiersin.org/articles/10.3389/fbloc.2021.789953.

A. Vaisman and E. Zimányi. Data-centric systems and applications data warehouse systems design and implementation second edition, 2022. URL https://bibliotecas.ups.edu.ec:3401/book/10.1007/978-3-662-65167-4.

Z. Zhang, Z. Liu, Q. Jiang, Z. Wu, J. Chen, and H. An. Rdma-based apache storm for high-performance stream data processing. In X. He, E. Shao, and G. Tan, editors, *Network and Parallel Computing*, pages 276–287, Cham, 2021. Springer International Publishing. ISBN 978-3-030-79478-1.

B. Zheng, X. Li, Z. Tian, and L. Meng. Optimization method for distributed database query based on an adaptive double entropy genetic algorithm. *IEEE Access*, 10:4640–4648, 2022. doi: 10.1109/ACCESS.2022.3141589.

6 Biografía

Jhony Cristian Llano Tumbaco. El autor es estudiante del programa de Maestría de *Software*. El obtuvo su título de Ingeniero en Software 2019 por la Universidad Politécnica Salesiana. Actualmente, se desempeña como: Coordinador de mantenimiento de desarrollo tecnológico en Corporación Favorita.
jllanot@est.ups.edu.ec

Paulina Adriana Morillo Alcívar. Obtuvo su grado de Ingeniería en Electrónica y Telecomunicaciones por la Escuela Politécnica Nacional (2012) y su posgrado en Gestión de la Información por la *Universitat Politècnica de València* (2016). Actualmente trabaja como profesora e investigadora en la Universidad Politécnica Salesiana (UPS). Posee experiencia en los campos de aprendizaje de máquina, matemática computacional y métodos numéricos.
pmorillo@ups.edu.ec