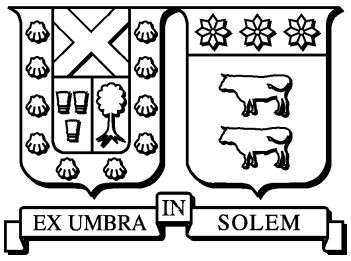
2018-07

ANÁLISIS DE CONSULTAS PREDICTIVAS EN BASES DE DATOS DE SERIES DE TIEMPO

COTORAS STRAUB, SLAVKO IVO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



ANÁLISIS DE CONSULTAS PREDICTIVAS EN BASES DE DATOS DE SERIES DE TIEMPO

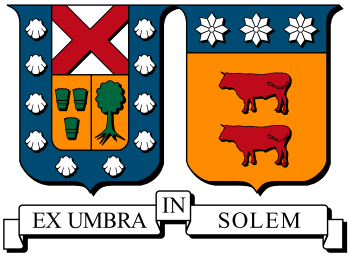
SLAVKO IVO COTORÁS STRAUB

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: JOSÉ LUIS MARTÍ LARA

JULIO 2018

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO – CHILE



ANÁLISIS DE CONSULTAS PREDICTIVAS EN BASES DE DATOS DE SERIES DE TIEMPO

SLAVKO IVO COTORÁS STRAUB

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA: JOSÉ LUIS MARTÍ LARA
PROFESOR CORREFERENTE: CECILIA REYES COVARRUBIAS

JULIO 2018

Agradecimientos

Agradezco a cada una de las personas que estuvieron presentes en el desarrollo de esta memoria, aportando con su alegría y compañía durante este desafío. En primer lugar agradezco a mi profesor guía José Luis Martí, quién con su apoyo y orientación, me orientó en el planteamiento del problema de investigación y desarrollo de mis experimentos. También agradezco a mi familia, quienes fueron el pilar fundamental para seguir adelante con este trabajo, en especial a mis padres Viviana e Ivo, y mis hermanos Branko y Dusan. Finalmente, agradecezco a mis amigos por sus palabras de aliento y optimismo en los momentos que más lo necesité.

Resumen

El presente trabajo muestra un análisis comparativo de alternativas de bases de datos de series de tiempo, orientado a usuarios que realizan consultas predictivas mediante el uso de una arquitectura de un solo nodo. El objetivo será identificar criterios al evaluar la precisión y rendimiento de los motores de bases de datos *InfluxDB*, *Graphite* y *RRDtool*, ejecutando un *benchmark* propio sobre dichos motores utilizando el mismo ambiente para su comparación.

De acuerdo al análisis que se realiza en esta memoria se concluye que si se prioriza la precisión en el modelo predictivo y/o el tiempo de ejecución de las consultas, se recomienda el uso de *RRDtool* y *Graphite*. Mientras que si se busca modificar los parámetros presentes en las consultas predictivas de forma nativa, se sugieren las bases de datos *InfluxDB* y *Graphite*; siendo estas, además, buenas opciones para la gestión remota de dichas consultas.

Palabras Claves: Series de tiempo, bases de datos, InfluxDB, Graphite, RRDtool, benchmark, Holt-Winters.

Abstract

This work shows a comparative analysis of time series database alternatives oriented to users that perform predictive queries through the use of a single node architecture. The objective will be to identify criteria when evaluating the accuracy and performance of the *InfluxDB*, *Graphite* and *RRDtool* database engines by running an own benchmark on them, using the same environment for comparison.

According to the analysis carried out in this report, it is concluded that if the precision in the predictive model and/or the execution time of the queries is prioritized, the use of *RRDtool* and *Graphite* is recommended. While if you want to modify the parameters present in predictive queries natively, the *InfluxDB* and *Graphite* databases are suggested; being these, in addition, good options for remote management of such queries.

Keywords: Time series, databases, InfluxDB, Graphite, RRDtool, benchmark, Holt-Winters.

Índice de Contenidos

Lista de Cuadros

Lista de Figuras

Introducción

La medición de valores a lo largo del tiempo es una práctica que hoy en día resulta ser común en la mayoría de las organizaciones y empresas, las cuales ven que los datos históricos les permiten identificar problemas y preveer posibles escenarios futuros. Esta sucesión de valores de una variable, la cual es medida y registrada en períodos iguales de tiempo, se conoce como serie de tiempo, donde el análisis de esta ocupa un papel protagónico en la predicción de un proceso o fenómeno.

Las bases de datos de series de tiempo almacenan dicha sucesión de valores, además de contar con una nueva manera para realizar pronósticos, siendo diferente a lo que tradicionalmente se desarrolla, como por ejemplo, el uso de un *software* externo a la base de datos que permite llevar a cabo una predicción. Las consultas predictivas son una forma de realizar una predicción, las cuales corresponden a una funcionalidad interna de la base de datos. Sin embargo, y dado que estas han surgido hace pocos años atrás, es que se tiene poco conocimiento de esta opción para realizar pronósticos.

En esta memoria se realiza un análisis de diferentes bases de datos de series de tiempo para conocer cómo funcionan y qué características poseen al ejecutar consultas predictivas, y así identificar los criterios que permitan establecer cuál de ellas es más adecuada bajo un entorno específico.

La estructura de la memoria se compone de cuatro capítulos que se describen a continuación. Capítulo 1, definición del problema junto a los objetivos y alcances de este trabajo. Capítulo 2, detalle de los estudios similares realizados anteriormente, en donde se incluye los distintos *benchmarks* que estos utilizan para el análisis, además de incluir la descripción

de diferentes bases de datos de series de tiempo existentes junto a la metodología, métodos y mediciones a seguir durante el desarrollo del experimento. Capítulo 3, fases del experimento y especificación de las herramientas a utilizar. Capítulo 4, construcción e implementación del entorno para desarrollar el experimento y la ejecución del mismo, en el cual se analizan los resultados obtenidos. Finalmente, en las conclusiones se lleva a cabo una discusión del tema desarrollado.

Capítulo 1

Definición del Problema

En este capítulo se presenta la motivación que da origen al tema a realizar en esta memoria, en donde se identifica el problema en cuestión, los objetivos planteados y el alcance para este trabajo.

1.1. Identificación del Problema

En la última década, se ha dado a conocer la importancia de almacenar los datos adquiridos por las empresas, ya sea al momento de entregar un servicio o bien al realizar alguna transacción, donde se registran los antecedentes del cliente y de lo que el mismo solicita a la empresa. Sin embargo, no es suficiente solo con recopilar los datos, puesto que si dicha acumulación de datos no es utilizada, no otorga ninguna utilidad siendo su almacenamiento injustificado y tórnandose en un costo sin beneficio para la empresa. Es por esto que identificar un motivo para almacenar dichos datos se vuelve trascendental para la empresa, dado que estas los aprovechan para su beneficio, fortaleciendo sus diferentes procesos, permitiendo por ejemplo compartir la información entre los departamentos existentes en una misma organización, y favoreciendo a una mejor comunicación interna lo que conlleva en un incremento del rendimiento laboral; por lo que es posible realizar los diferentes trabajos de una forma más sencilla, rápida y eficaz, lo que se traduce en un ahorro en el tiempo estimado de ejecución

de cada labor, y por ende, una disminución en el costo de realizar la misma. Además, dada la facilidad que presenta disponer del uso de la información, es posible mejorar la calidad de la relación con el cliente puesto que se pueden ofrecer respuestas a sus necesidades de una forma más rápida. Por otra parte, disponer de datos históricos de una empresa permite que estos puedan ser utilizados para complementar la solución a uno de los temas que con frecuencia es analizado por las empresas, la cual corresponde a la incertidumbre dada por lo que ocurrirá con posterioridad.

Según *Oxford Dictionaries*, una predicción se define como: "hecho o situación que se anuncia que sucederá en el futuro" [2], como por ejemplo, el número de ventas en un día de festividad o el flujo de clientes que son atendidos en una sucursal. Esto permite que la empresa mejore su accionar ya que se tiene conocimiento de lo que puede ocurrir, permitiendo así que la toma de decisiones sea con un menor riesgo e incertidumbre, a partir del comportamiento histórico de la empresa.

Existen diferentes métodos para establecer pronóstico, los cuales se dividen en dos tipos clásicos: cualitativos y cuantitativos. En el primer tipo de predicción se prioriza la subjetividad, donde se valoran la experiencia y opinión al momento de generar un pronóstico, siendo su uso clave al momento de no contar con datos históricos, o si estos son escasos. Por otro lado, los métodos cuantitativos estiman los valores futuros de la variable de interés mediante un estudio de los datos históricos; dentro de estos métodos, se encuentran las denominadas series de tiempo, secuencias de datos recopiladas en intervalos regulares de tiempo. De esta manera, se genera la necesidad en las empresas de generar pronósticos aprovechando la facilidad que se presenta en el manejo de los datos históricos de las series temporales, siendo las bases de datos de series de tiempo una buena opción ante dicho requerimiento.

Las bases de datos de series de tiempo son una colección de series de tiempo, la cual se encuentra optimizada para la manipulación de estos datos debido a que se encuentran indexadas por el tiempo. A pesar de que las bases de datos de series de tiempo surgieron hace ya tiempo atrás, no se había generado un interés importante en las empresas. Sin embargo, debido a la fuerte aparición del internet de las cosas y el auge en que se encuentra la denominada *Big Data*, donde los diferentes dispositivos tecnológicos generan una gran cantidad de datos que son almacenados en grandes bases de datos y posteriormente utilizados para el análisis, se ha

percibido un crecimiento en el interés de las bases de datos de series de tiempo por sobre los demás tipos de bases de datos, fenómeno que se refleja en la figura 1.1, donde *DB-Engines* presenta la tendencia de la popularidad de cada tipo de base de datos, entre junio del 2016 y junio del 2018, métrica que considera tanto el uso de las bases de datos, como también las búsquedas, preguntas y referencias que se han realizado de cada una de ellas.

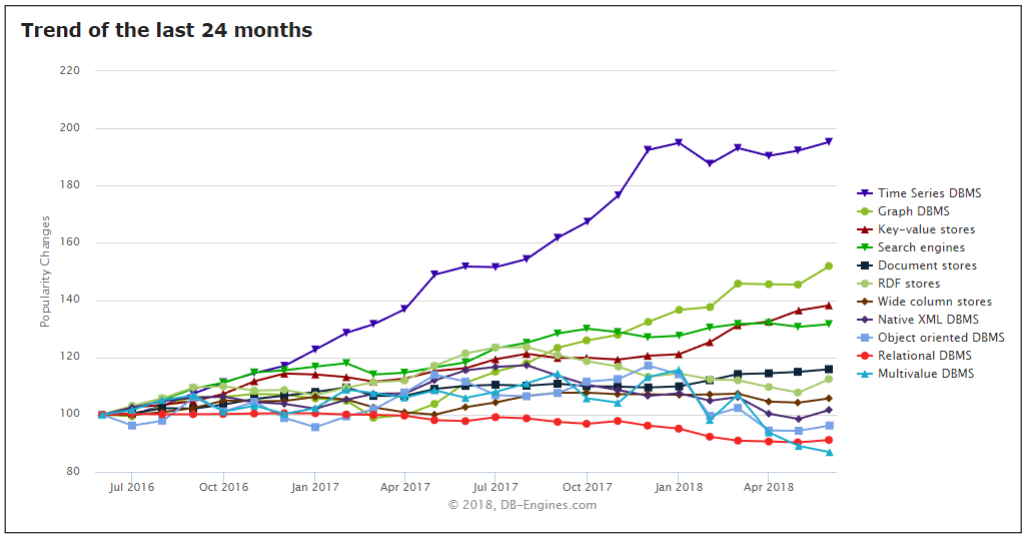


Figura 1.1: Tendencia de la puntuación de popularidad propuesto por *DB-Engines* de los diferentes tipos de bases de datos, obtenidos entre junio del 2016 y junio del 2018 [74].

Dicha tendencia se ha traducido en el nacimiento de diferentes motores de bases de datos de serie de tiempo, los cuales han integrado las predicciones mediante el uso de consultas predictivas, evitando así la necesidad de uso de *software* de terceros para generar pronósticos. Esta memoria pretende contar con un marco que permita identificar cuál(es) motores de bases de datos de series de tiempo es(son) más adecuado(s) para un escenario dado, a partir de la evaluación de algunos motores considerando factores de precisión y rendimiento en los resultados de las predicciones a realizar.

1.2. Objetivo General

Realizar un estudio comparativo de consultas predictivas de bases de datos de series de tiempo, midiendo la precisión y rendimiento de tres motores de dicho tipo, para contar con un

marco que permita establecer cuál es el más adecuado bajo un entorno específico.

1.3. Objetivos Específicos

- Revisar el estado del arte sobre motores de bases de datos de series de tiempo, para identificar los más utilizados, además de sus características y requisitos.
- Realizar la comparación de los tres motores de bases de datos de series de tiempo seleccionados, con la finalidad de contrastar la precisión y rendimiento de las predicciones en distintos escenarios de prueba.
- Recomendar al lector qué base(s) de datos de series de tiempo es(son) la(s) más adecuada(s) para su implementación y su nivel de precisión en las predicciones, a fin de elegir la mejor alternativa, según el entorno presentado.

1.4. Alcance

- Se trabaja con software *open source*, es por esto, que las pruebas se realizan con una arquitectura de un solo nodo debido a la limitante presente en *InfluxDB*, el cual requiere versión empresarial para hacer uso de múltiples nodos [34].
- Los motores de bases de datos de series de tiempo a considerar en la experimentación son aquellos que cuenten con funciones de predicción propias; los que requieran funciones externas para realizar una predicción no son incluidas en este trabajo.
- Los motores de bases de datos a considerar en la experimentación son aquellos que estén catalogados como bases de datos de series de tiempo según *DB-Engines*, lo que deja afuera a motores como *Kdb+* el cual presenta un multi-modelo.
- Los *sets* de datos con los cuales se realizan las pruebas corresponden a los presentes en *UCI Machine Learning Repository* [52], repositorio que se encuentra disponible para uso público.

Capítulo 2

Estado del Arte

En este capítulo, se describen los conceptos generales relacionados con las series temporales y métodos predictivos, y se detallan algunos de los estudios que se han realizado en el ámbito de las bases de datos de series de tiempo. Posteriormente, se realiza una revisión de las bases de datos de series de tiempo que son más utilizadas en la actualidad, de modo que el lector pueda tener una mejor perspectiva del trabajo.

2.1. Predicción

Predicción se define como una "conjetura de algo que ha de suceder" [1], donde se pueden encontrar diferentes tipos de pronósticos. Por una parte, los pronósticos cualitativos, los cuales se basan en técnicas subjetivas mediante el juicio personal y el uso de cualidades como la intuición, experiencia y la opinión experta [79], siendo el método Delphi una de lás técnicas más utilizadas [68]. Por otra parte, se encuentran los pronósticos cuantitativos, en los cuales se usan dos metodologías basadas en estadísticas convencionales: (1) análisis de series de tiempo y (2) de regresión. Una serie de tiempo es un conjunto de valores secuenciales en el tiempo, mientras que un análisis de regresión es una técnica que se basa en las observaciones que se realizan a cada variable, la cual es utilizada para estudiar la relación entre ellas, dando

así a conocer una ecuación matemática de dicha correspondencia [97]. Los sistemas inteligentes utilizan una combinación entre las técnicas cuantitativas y cualitativas, lo que permite que variables con tendencias fluctuantes puedan ser evaluadas [79].

En este trabajo, se profundiza la investigación en los pronósticos cuantitativos, donde se enfoca el estudio en las predicciones mediante el análisis de series de tiempo.

2.2. Series de Tiempo

Una serie de tiempo se define como un *"colección o conjunto de mediciones de cierto fenómeno o experimento registrados secuencialmente en el tiempo, en forma equiespaciada"* [47]. Cada serie de tiempo está compuesta por tres componentes básicos, los cuales son detallados a continuación [6]:

1. **Tendencia**: existe tendencia cuando a largo plazo se presenta un incremento o disminución gradual en los datos, esta puede no ser lineal.
2. **Estacionalidad**: existe estacionalidad cuando una serie está influenciada por factores repetitivos a corto plazo.
3. **Residual**: se conoce como residual o irregularidad al componente que presenta un aumento o disminución aleatoria en los datos durante un período específico de tiempo.

Cada componente de una serie de tiempo se refleja en la figura 2.1, la cual muestra primero una serie temporal completa.

En algunas investigaciones, se considera una cuarta componente la cual corresponde a la componente cíclica. Esta consiste en que ante cambios graduales en los datos a largo plazo, estos suben y bajan de manera irregular [12].

En la desagregación de una serie temporal, se consideran dos descomposiciones básicas que dan origen a una serie de tiempo: (1) aditiva y (2) multiplicativa. La descomposición aditiva es aquella que permite la composición de una serie de tiempo mediante la suma de cada

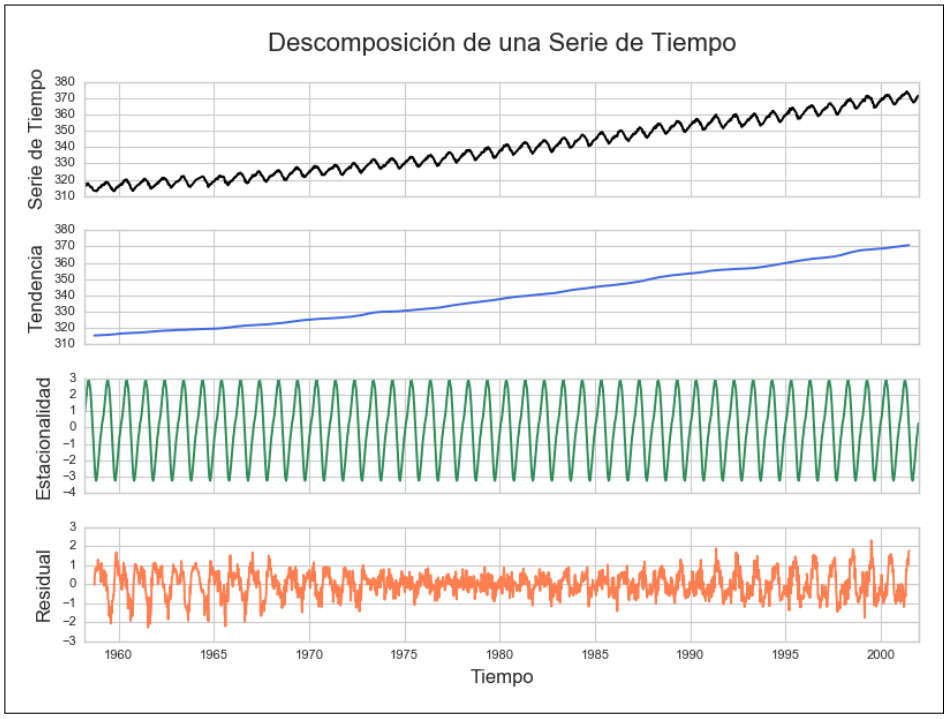


Figura 2.1: Descomposición de una serie de tiempo (elaboración propia).

componente básico de una serie temporal. Este modelo de descomposición es frecuentemente usado cuando la variación del componente de estacionalidad es relativamente constante en el tiempo [59]. La representación matemática de este modelo está dada por la ecuación 2.1, donde T_t, E_t y R_t corresponden a los componentes típicos de una serie de tiempo: tendencia, estacionalidad y residual, respectivamente.

$$x_t = T_t + E_t + R_t \tag{2.1}$$

Por otra parte, la descomposición multiplicativa se consigue mediante el producto entre cada uno de los componentes de una serie de tiempo. Es conveniente utilizar este modelo cuando la variación de estacionalidad incrementa a través del tiempo [59]. Su representación matemática está dada por la ecuación 2.2, donde el significado de los términos es equivalente al usado en la ecuación 2.1.

$$x_t = T_t \cdot E_t \cdot R_t \tag{2.2}$$

La principal diferencia entre ambos modelos de descomposición se encuentra en su componente de estacionalidad. Si se observa en la figura 2.2, mientras el modelo aditivo presenta un comportamiento estacional constante en el tiempo, el modelo multiplicativo aumenta la magnitud de dicha componente en el transcurso del tiempo.

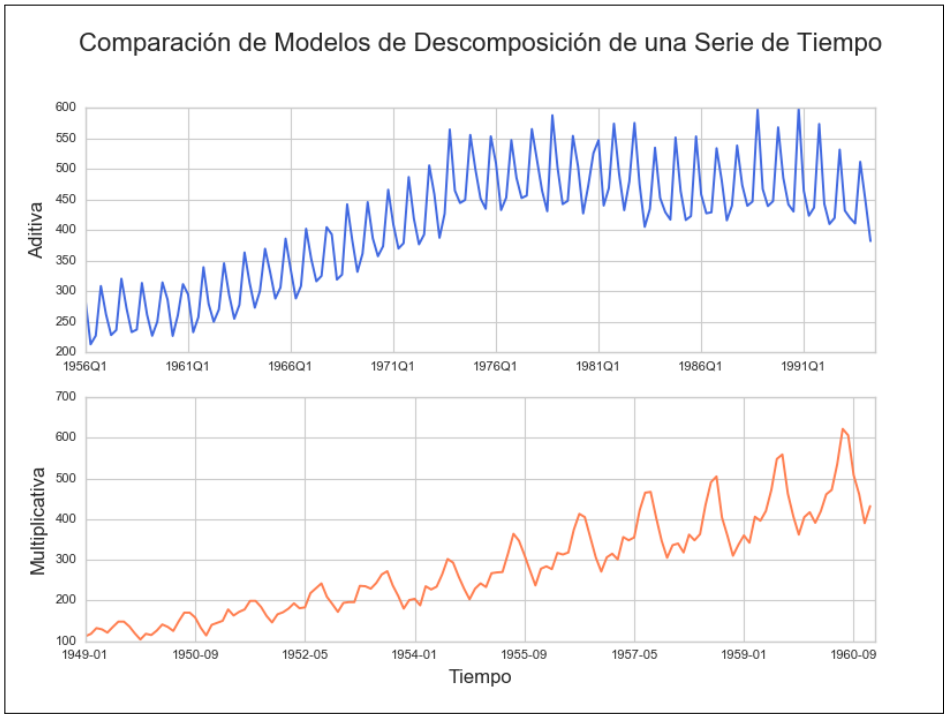


Figura 2.2: Comparación de modelos de descomposición de una serie de tiempo (elaboración propia).

2.3. Métodos Predictivos

En esta sección se describen los métodos de predicción de datos en series de tiempo más utilizados en la actualidad.

2.3.1. Modelo ARIMA

En el año 1976 tras la publicación del libro, "*Time Series Analysis, Forecasting and Control.*", escrito por Box y Jenkins [7], se describe una metodología para analizar y pronosticar datos de series de tiempo conocida como modelo ARIMA. El nombre deriva de sus tres componentes [66]: autorregresivo (AR), integrado (I) y medias móviles (MA). La primera componente del modelo permite capturar la variación histórica de los datos reales a un modelo de pronóstico, y utiliza dichos valores para crear un mejor modelo de predicción. La componente "integrada" incorpora al modelo la cantidad de diferenciación para aplicar a la serie de tiempo, es decir, el número de valores en períodos previos a restar del valor actual para ser aplicados en la serie temporal. La última componente del modelo, "medias móviles", permite dar a conocer el error en el modelo como una combinación lineal de los valores de error observados en los puntos de períodos previos de tiempo.

Dicho modelo es una extensión del modelo ARMA, el cual se compone de dos partes: autorregresivo (AR) y medias móviles (MA). El modelo autorregresivo pronostica la variable objetivo usando una combinación lineal de valores pasados de la variable. En la ecuación 2.3 se define la expresión matemática de un modelo autorregresivo de orden p [23], donde c corresponde a una constante y e_t es un factor de ruido blanco [60]; al variar los parámetros de $\phi_1, ..., \phi_p$ se obtienen diferentes patrones de series de tiempo.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t \tag{2.3}$$

Por otra parte, el modelo de medias móviles utiliza los errores de predicción en una regresión en vez de los valores pasados como se hace en el modelo anterior; la expresión de este modelo con orden q está dada por la ecuación 2.4 [29], siendo similar a la presentada en el modelo pasado, donde la variación en los parámetros de $\theta_1, ..., \theta_q$ dan como resultado diferentes patrones de series de tiempo.

$$y_t = c + \theta_1 y_{t-1} + \theta_2 y_{t-2} + \cdots + \theta_q y_{t-q} + e_t \tag{2.4}$$

El modelo ARIMA combina el modelo de autorregresión con el modelo de medias móviles, y además incorpora la diferenciación, lo que da origen a su acrónimo de ARIMA, la cual deriva de "*autoregressive integrated moving average*". Una serie de diferenciación se define como el cambio entre observaciones consecutivas en la serie original [32], y está dada por la expresión: $y'_t = y_t - y_{t-1}$, donde la serie diferenciada tendrá $t - 1$ datos, puesto que no es posible el cálculo para la primera observación. Esta diferenciación se puede dar también para casos de estacionalidad, en el cual la diferencia se realiza entre una observación y la correspondiente al año anterior; su expresión está dada por: $y'_t = y_t - y_{t-m}$, donde m corresponde al número de estaciones presentes en un año. El modelo ARIMA permite describir un dato de la serie de tiempo como una función lineal de valores anteriores y errores debidos al azar [72]. La expresión matemática del modelo ARIMA está definida en la ecuación 2.5, donde y'_t es la serie de diferenciación y d corresponde al grado de diferenciación. Esta expresión matemática se denota como modelo $ARIMA(p, d, q)$, o también como modelo ARIMA no estacional [30].

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t \tag{2.5}$$

Para trabajar con modelos más complejos al combinar los componentes del modelo ARIMA, se recomienda el uso de la notación *backshift* [24], la que permite un fácil manejo de los datos de las series de tiempo en períodos anteriores; de esta manera, un retardo en un valor de la serie temporal se escribe de la forma: $B\,y_t = y_{t-1}$. Además, el número de desplazamientos hacia atrás de un dato está definido por el exponente, así, si se quiere desplazar los datos hacia dos períodos anteriores, la notación será de la forma:

$$B\,(B\,y_t) = B^2\,y_t = y_{t-2}$$

Esta notación es útil al ser combinada con otras diferencias, dado que este operador puede tratarse usando las reglas algebraicas ordinarias.

Al integrar la notación *backshift* en el modelo ARIMA, en la ecuación 2.5, queda definida de la forma presente en la ecuación 2.6.

$$\underbrace{(1-\phi_1 B-\cdots-\phi_p B^p)}_{AR(p)}\ \underbrace{(1-B)^d y_t}_{\text{d diferencias}} \quad = \quad \underbrace{c+(1+\theta_1 B+\cdots+\theta_q B^q)e_t}_{MA(q)} \tag{2.6}$$

El modelo ARIMA no solo se restringe a los datos de series de tiempo no estacionales, sino que también permite modelar datos de series estacionales. Un modelo ARIMA permite ser estacional al incluir términos estacionales adicionales en dicho modelo. De esta manera, el modelo ARIMA se escribe como se muestra en la figura 2.3 [31], donde m corresponde a la cantidad de períodos por temporada (se usa notación en minúsculas para las partes no estacionales del modelo, y notación en mayúsculas para aquellas partes estacionales de dicho modelo).

$$ARIMA\ \underbrace{(p,d,q)}_{\substack{\uparrow \\ \text{Parte} \\ \text{No Estacional}}}\ \underbrace{(P,D,Q)_m}_{\substack{\uparrow \\ \text{Parte} \\ \text{Estacional}}}$$

Figura 2.3: Modelo ARIMA Estacional (elaboración propia).

La parte estacional del modelo utiliza términos muy similares a los componentes no estacionales, sin embargo, emplea períodos previos relacionados con la estacionalidad. La combinación de parte estacional y no estacional de dicho modelo se conoce como modelo ARIMA estacional, o bien SARIMA, el cual deriva de "*seasonal autoregressive integrated moving average*" [46]. Así, para ejemplificar: un modelo $ARIMA(1,1,1)(1,1,1)_4$, donde no se considera la constante y $m = 4$, valor usado para estacionalidad de los datos trimestrales, la expresión matemática queda definida por la ecuación 2.7.

$$\overbrace{(1-\phi_1 B)}^{\substack{\text{AR(1)} \\ \text{No Estacional}}}\underbrace{(1-\Phi_1 B^4)}_{\substack{\text{AR(1)} \\ \text{Estacional}}}\ \overbrace{(1-B)}^{\substack{\text{Diferenciación} \\ \text{No Estacional}}}\ \underbrace{(1-B^4)}_{\substack{\text{Diferenciación} \\ \text{Estacional}}}\ y_t = \overbrace{(1+\theta_1 B)}^{\substack{\text{MA(1)} \\ \text{No Estacional}}}\underbrace{(1+\Theta_1 B^4)}_{\substack{\text{MA(1)} \\ \text{Estacional}}} e_t \tag{2.7}$$

2.3.2. Suavizamiento Exponencial Simple

Se denomina suavizamiento exponencial a los promedios ponderados de observaciones pasadas, haciendo que el peso dacaiga de forma exponencial a medida que los datos envejecen; de esta manera, se le otorga un mayor peso asociado a las observaciones más recientes [25]. El suavizamiento exponencial simple es la forma general de este tipo de métodos predictivos, el cual se utiliza en aquellas series de tiempo que presentan un patrón horizontal, sin alguna presencia de tendencia o estacionalidad. El único parámetro de suavizado de este método es el nivel, componente dinámico individual presente en un modelo de promedio móvil. Este modelo suaviza los datos al promediar los valores más recientes en una serie. La expresión matemática de este método está dada en la ecuación 2.8, la cual se basa en los valores de la predicción y el real observado del período anterior para realizar el pronóstico del período siguiente, donde y_t e y_{t+1} corresponden al valor de predicción para el período t y $t+1$, respectivamente; X_t es el valor real observado en el tiempo t; y α y $(1-\alpha)$ representan a factores de ponderación.

$$y_{t+1} = \alpha X_t + (1-\alpha) y_t \tag{2.8}$$

Los valores que pueden tomar los factores de ponderación α y $(1-\alpha)$ se encuentran entre 0 y 1, y es lo que permite la atenuación de los datos en la serie de tiempo.

2.3.3. Método de Holt

En el año 1957, Holt extiende el suavizamiento exponencial simple mediante el método de Holt o doble suavizamiento exponencial, el cual permite la predicción de datos con una tendencia, por lo que este método es utilizado en series de tiempo con tendencia lineal que no presenten estacionalidad. Los parámetros de suavizado de este método son el nivel y la tendencia. Este método utiliza una ecuación de pronóstico dada por la ecuación 2.9, y dos ecuaciones de suavizado, una para cada parámetro de suavizado, las cuales estan definidas en las ecuaciones 2.10 y 2.11 [26], donde y_{t+h} corresponde al pronóstico para el período $t+h$,

ℓ_t es la estimación del nivel de la serie en el tiempo t, b_t es la estimación de la tendencia de la serie de tiempo en el período t, y β^* denota el parámetro de suavizado para la tendencia, el cual toma valores entre 0 y 1.

$$
\begin{aligned}
y_{t+h} &= \ell_t + hb_t && (2.9)\\
\ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) && (2.10)\\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} && (2.11)
\end{aligned}
$$

De igual modo al método de suavizamiento exponencial simple, la ecuación de nivel muestra que ℓ_t es un promedio ponderado entre la observación y_t y la predicción dentro de la muestra de un período posterior al tiempo t, denotadas por $\ell_{t-1} + b_{t-1}$. Por otra parte, la ecuación de tendencia muestra que b_t es un promedio ponderado de la tendencia estimada en el período t, dado por $\ell_t - \ell_{t-1}$ y la estimación previa de la tendencia b_{t-1}. La función de predicción deja de tener un patron horizontal, pues ahora presenta un comportamiento de tendencia, donde h corresponde a los períodos posteriores de predicción respecto al último valor estimado de la tendencia; de esta manera los pronósticos son una función lineal de h.

2.3.4. Método de Holt-Winters

Holt y Winters extienden el método de Holt para capturar la componente estacional de la serie de tiempo; de esta manera el método Holt-Winters, o suavizamiento exponencial triple, permite la predicción de datos cuando hay tendencia y estacionalidad, agregando un tercer parámetro de suavizado el cual considera la estacionalidad. Este método considera una ecuación de predicción y tres de suavizado [27]: una para el nivel ℓ_t, otra para la tendencia b_t y una tercera para la componente estacional denotado por s_t, con parámetros de suavizado estan dados por α, β^* y γ, respectivamente. Además, se usa m, que corresponde al período de la estacionalidad, es decir, el número de estaciones presentes en un año y el factor h_m^+, dado por $\lfloor (h-1) \; mod \; m \rfloor + 1$, la cual garantiza que las estimaciones de los índices estacionales utilizados para la predicción proceden del último año de la muestra.

El método de Holt-Winters presenta dos variaciones, las cuales difieren según la naturaleza del componente estacional: (1) método aditivo y (2) multiplicativo. La primera variación es recomendada cuando las variaciones estacionales son relativamente constantes a través de los datos de la serie. Además, la componente estacional de dicha variación es expresada en valores absolutos en la escala de la serie observada, y en la ecuación de nivel la serie se ajusta estacionalmente al restar la componente estacional en cada período. Es así como al cabo de un año, el valor de la suma de cada componente estacional será de aproximadamente cero. La expresión matemática del pronóstico del método aditivo está dada por la ecuación 2.12, y las ecuaciones de suavizado de nivel, tendencia y estacionalidad están definidas por las ecuaciones 2.13, 2.14 y 2.15, respectivamente.

$$y_{t+h} = \ell_t + hb_t + s_{t-m+h_m^+} \tag{2.12}$$

$$\ell_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \tag{2.13}$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \tag{2.14}$$

$$s_t = \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{2.15}$$

La ecuación de nivel muestra un promedio ponderado entre la observación ajustada estacionalmente $(y_t - s_{t-m})$ y el pronóstico no estacional $(\ell_{t-1} + b_{t-1})$ en el período t. Por otro lado, la ecuación de tendencia es la misma a la que se utiliza en el método de Holt, mientras que la ecuación estacional muesta un promedio ponderado entre el índice estacional actual, $(y_t - \ell_{t-1} - b_{t-1})$ y el índice estacional de la misma temporada del año pasado.

Por otra parte, el método multiplicativo se recomienda cuando las variaciones estacionales cambian proporcionalmente al nivel de la serie. Además, la componente estacional se expresa en términos relativos, es decir en porcentajes, y los datos de la serie se ajustan estacionalmente dividiéndose por el componente estacional. La ecuación de predicción del método multiplicativo está dada por la ecuación 2.16, y las ecuaciones de suavizado del nivel, la tendencia y estacionalidad están definidas por las ecuaciones, 2.17, 2.18 y 2.19, respectivamente.

$$y_{t+h} = (\ell_t + hb_t)s_{t-m+h_m^+} \quad (2.16)$$

$$\ell_t = \alpha\frac{y_t}{s_{t-m}} + (1-\alpha)(\ell_{t-1} + b_{t-1}) \quad (2.17)$$

$$b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \quad (2.18)$$

$$s_t = \gamma\frac{y_t}{\ell_{t-1} - b_{t-1}} + (1-\gamma)s_{t-m} \quad (2.19)$$

2.4. Medición del Error

La medición del error corresponde a la diferencia entre el valor medido y el valor real observado. En esta sección, se describen diferentes métricas que son utilizadas en la medición del error de predicción de las series de tiempo.

2.4.1. Error Absoluto Medio (MAE)

El error absoluto medio mide la magnitud promedio de los errores en un conjunto de predicciones, sin tomar en cuenta sus direcciones. Se define como el promedio de las diferencias absolutas entre la predicción y el valor real observado, en el que cada diferencia individual es del mismo peso que el resto [44]. En la ecuación 2.20 se representa esta medida, donde x corresponde al valor real, y es el valor de la predicción, y n es el tamaño del conjunto de datos en observación.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - x_j\right| \quad (2.20)$$

Este error presenta una limitante al ser dependiente de la escala de los datos, por lo cual, al momento de comparar conjuntos de predicciones, estos deben ser de la misma escala de tiempo para que tenga sentido el uso de esta medida.

2.4.2. Error Porcentual Absoluto Medio (MAPE)

El error porcentual absoluto medio mide el porcentaje de error en un conjunto de predicciones. Se define como el promedio de la diferencia entre el valor real observado y la predicción, expresado como porcentaje de la observación real [51]. Su representación matemática esta dada por la ecuación 2.21.

$$MAPE = \sqrt{\frac{100}{n}\sum_{j=1}^{n}\left|\frac{x_j - y_j}{x_j}\right|} \tag{2.21}$$

Al ser un error porcentual, tiene la ventaja de ser independiente de la escala de tiempo, por lo que es frecuentemente usado para comparar el conjunto de predicciones entre diferentes *sets* de datos. Sin embargo, posee la desventaja de indefinirse si un valor de la observación real es cero. Otro inconveniente que se presenta al utilizar errores porcentuales es que pierde el sentido del error al comparar *sets* de datos con valores negativos, por ejemplo, el pronóstico de temperatura, dado que se asumen valores como cero significativo [28].

2.4.3. Error Cuadrático Medio (RMSE)

El error cuadrático medio es una regla de puntuación cuadrática que mide la magnitud promedio de los errores en un conjunto de predicciones. Se define como la raíz cuadrada del promedio de las diferencias al cuadrado entre la predicción y el valor real observado [10]. Su representación matemática está dada por la ecuación 2.22.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_j - x_j\right)^2} \tag{2.22}$$

Esta medición tiene una relación directa con el coeficiente de correlación. Cuando el valor de dicho coeficiente es 1, existe una fuerte correlación entre los datos comparados, por lo que el valor de esta medida de error será 0, debido a que todos los puntos se encuentran en la línea de regresión, es decir, no hay presencia de error entre los datos a comparar [4].

2.4.4. Error Absoluto Medio Escalado (MASE)

El error absoluto medio escalado propone escalar el error basado en el denominado error absoluto medio, el cual hace uso del método de predicción ingenuo que permite generar un pronóstico de un período adelante de cada elemento presente en el conjunto de datos de la muestra. De esta manera, el error escalado q_t se define por la ecuación 2.23.

$$q_t = \frac{y_j - x_j}{\frac{1}{n-1}\sum_{j=2}^{n}\left|y_j - y_{j-1}\right|} \tag{2.23}$$

El valor del error escalado será menor a 1 si la predicción es mejor que el promedio del pronóstico ingenuo de cada elemento calculado de la muestra; por otra parte, su valor será mayor a 1 al tener una predicción peor a la obtenida del promedio del pronóstico ingenuo [69]. Es así, como el error absoluto medio escalado se define como el promedio absoluto de los errores escalados calculados de cada elemento presente en la muestra, y su representación matemática está dada por la ecuación 2.24.

$$MASE = mean(|q_t|) \tag{2.24}$$

Esta medición es independiente de la escala de los datos, lo cual permite ser una ventaja al momento de comparar *sets* de datos, puesto que no requiere que los datos de dichos conjuntos se encuentren en la misma escala. Por otro lado, esta medida de error permite ser una alternativa al uso de los errores porcentuales, dado que este error evita indefinirse ante la presencia del valor cero en la observación real.

2.5. Bases de Datos de Series de Tiempo (TSDB)

Las bases de datos de series de tiempo corresponden a un sistema de gestión de bases de datos, las cuales han sido optimizadas para el manejo de datos de series de tiempo, donde cada entrada de dato es asociada a una marca de tiempo [76].

Entre las bases de datos de series de tiempo existentes, *DB-Engines* ha generado un ranking mediante una propuesta de puntuación de popularidad de cada una de ellas; dicho ranking se encuentra presente en el cuadro 2.1, donde se incluyen los motores de bases de datos: *InfluxDB*, *Graphite*, *RRDtool*, *OpenTSDB* y *Prometheus*, los que son descritos en esta sección.

Rank			DBMS	Database Model	Score		
Jun 2018	May 2018	Jun 2017			Jun 2018	May 2018	Jun 2017
1.	1.	1.	InfluxDB	Time Series DBMS	11.33	+0.33	+3.13
2.	2.	↑ 5.	Kdb+	Multi-model	3.02	-0.06	+1.44
3.	3.	↓ 2.	RRDtool	Time Series DBMS	2.67	-0.01	-0.35
4.	4.	↓ 3.	Graphite	Time Series DBMS	2.38	+0.12	+0.38
5.	5.	↓ 4.	OpenTSDB	Time Series DBMS	1.56	-0.06	-0.24
6.	6.	↑ 8.	Prometheus	Time Series DBMS	1.27	+0.14	+0.66
7.	7.	↓ 6.	Druid	Time Series DBMS	1.13	+0.12	+0.13
8.	8.	↓ 7.	KairosDB	Time Series DBMS	0.41	-0.02	-0.21
9.	9.	9.	eXtremeDB	Multi-model	0.28	-0.03	-0.09
10.	10.	↑ 11.	Riak TS	Time Series DBMS	0.21	-0.05	-0.03
11.	↑ 14.	↓ 10.	Axibase	Time Series DBMS	0.11	+0.06	-0.14
12.	↓ 11.	↑ 14.	FaunaDB	Multi-model	0.11	+0.00	+0.02
13.	↓ 12.	↑ 19.	Hawkular Metrics	Time Series DBMS	0.11	+0.00	+0.07
14.	↓ 13.	↑ 15.	Blueflood	Time Series DBMS	0.09	-0.01	+0.01
15.	↑ 16.	↓ 12.	Warp 10	Time Series DBMS	0.06	+0.02	-0.14
16.	↑ 17.		TimescaleDB	Time Series DBMS	0.05	+0.01	
17.	↓ 15.	↑ 18.	Quasardb	Time Series DBMS	0.04	-0.01	-0.01
18.	18.	↓ 13.	TempoIQ	Time Series DBMS	0.02	-0.00	-0.17
19.	↑ 20.	↑ 21.	Heroic	Time Series DBMS	0.00	±0.00	-0.02
19.	↑ 20.		IRONdb	Time Series DBMS	0.00	±0.00	
19.	19.	↓ 17.	Machbase	Time Series DBMS	0.00	-0.01	-0.06
19.	↑ 20.	↑ 22.	Newts	Time Series DBMS	0.00	±0.00	±0.00
19.	↑ 20.	↑ 22.	SiriDB	Time Series DBMS	0.00	±0.00	±0.00
19.	↑ 20.	↓ 16.	SiteWhere	Time Series DBMS	0.00	±0.00	-0.06
19.	↑ 20.	↑ 20.	Yanza	Time Series DBMS	0.00	±0.00	-0.03

Cuadro 2.1: Ranking de motores de bases de datos de series de tiempo según el puntaje de popularidad propuesto por *DB-Engines*[1] [73].

[1]Los valores presentados corresponden al puntaje de popularidad calculados hasta la fecha 12 de junio del año 2018.

2.5.1. InfluxDB

InfluxDB [56] es una base de datos de series de tiempo *open source* con opción comercial para la escalabilidad y *clustering*. Está escrita en *Go* y no tiene dependencia de ningún otro sistema de gestion de bases de datos. *InfluxDB* está orientado a ser utilizado como un respaldo para casos que involucren grandes cantidades de datos indexados por el tiempo, como son las métricas de aplicaciones, datos de sensores utilizados para el internet de las cosas y análisis en tiempo real. Previo a la inserción de datos, se debe crear una base de datos, la cual acepta diferentes atributos para ser insertadas en dicha base de datos, sin necesidad de crear cada una previamente. *InfluxDB* usa dos protocolos para insertar datos a la base de datos: (1) un protocolo basado en texto llamado "*Line Protocol*" y (2) un protocolo obsoleto de JSON [40].

Cada consulta de *InfluxDB* tiene una sintaxis similar a la presente en las consultas SQL, siendo *InfluxDB Query Language* (*InfluxQL*) el lenguaje de dichas consultas. Dispone de un amplio soporte en funciones de agregación, donde se encuentran las funciones: *count*, *mean*, *median* y *sum* [39]. Además cuenta con la función de predicción *holt_winters*, la cual permite predecir un número definido de datos mediante el uso del método Holt-Winters, a explicar en la sección 2.5.4.

InfluxDB posee una granularidad para los datos almacenados de hasta un nanosegundo [43]. Puede lograr alta disponibilidad, balanceo de cargas y escalabilidad mediante el uso de más de una instancia del motor de base de datos. Sin embargo, el uso de uno o más instancias como *cluster* se encuentra restringido por la opción comercial, la cual requiere de la versión empresarial para permitir dicha distribución [37]. La interfaz usuaria que proporciona *InfluxDB* es mediante una interfaz *HTTP* y de linea de comando, denominada cliente *Command Line Interface* (CLI) [41]. Se provee de interfaces para consultas a través de interfaces de *HTTP*, *UDP*, *OpenTSDB*, *Graphite* y *Collectd* [42]. Además, *InfluxDB* cuenta con un amplio soporte en bibliotecas de cliente, dentro de las cuales se pueden encontrar soporte para lenguajes como *Python*, *R*, *Ruby*, *Java* y *Go* [36], facilitando el uso de su *API*.

2.5.2. Graphite

Graphite [20] es una herramienta *open source* de monitoreo, que almacena datos numéricos de series de tiempo. *Graphite* consta de tres componentes de *software*: (1) *carbon*, (2) *whisper* y (3) *graphite webapp*. *Carbon* es un *twisted daemon* [92] que escucha datos de series de tiempo; *whisper* es una biblioteca de base de datos simple para almacenar datos de series temporales, la cual tiene un diseño y propósito similar a la *Round Robin Database* (*RRD*), en donde la base de datos mantiene un tamaño fijo y permite la retención de datos a largo plazo mediante la degradación de la granularidad de los datos históricos con mayor antigüedad. Finalmente, *graphite webapp* consiste en una aplicación web de *Django* [13] que permite procesar gráficos bajo demanda utilizando la biblioteca gráfica *Cairo* [9]. De esta manera, las aplicaciones cliente envían flujos de datos de series temporales al componente *carbon* de *Graphite*, la cual almacena dichos datos en archivos de base de datos *whisper*. Luego la componente *graphite webapp* proporciona la interfaz web que permite visualizar mediante gráficos los datos almacenados, así como también dispone de una *API* basada en URL simple para la generación directa de gráficos, llamada *Render*.

Graphite cuenta con dos protocolos para insertar datos en la base de datos: (1) protocolo de texto plano y (2) protocolo de *pickle*. El primero envía los datos con un formato específico, sobre el cual el componente *carbon* permite traducir la linea de texto en una métrica para ser almacenada en la base de datos *whisper*. Por otro lado, el protocolo de *pickle* es una opción más eficiente que el anterior dado que admite lotes de métricas que son enviadas a la componente *carbon* de una sola vez. La granularidad para los datos almacenados en la componente *whisper* de *Graphite* es de hasta un segundo. *Graphite* soporta el uso de funciones de agregación, entre las que se encuentran funciones como: *average*, *median*, *sum* y *count* [18]. Además, mediante el uso de la función *holtWintersForecast*, permite predecir datos con el método Holt-Winters.

Graphite puede lograr escalabilidad y alta disponibilidad tras la incorporación de nuevas máquinas para obtener un mayor rendimiento. *Carbon* proporciona la capacidad para realizar un balanceo de cargas. Por otra parte, *Graphite* cuenta con bibliotecas de cliente con soporte para *Node.js*, *Javascript* y *Python* [19].

2.5.3. RRDtool

RRDtool [86] es un sistema *open source* que permite almacenar y mostrar datos de series de tiempo, siendo uno de los más antiguos en este campo. Usa *RRD* para el almacenamiento de los datos de un rango de tiempo específico, por ejemplo, los datos de un año. De esta manera, el tamaño de la base de datos no crece con el tiempo, dado que siempre mantiene un tamaño determinado, eliminando los datos más antiguos que se encuentren fuera del rango específico de tiempo. Además, al utilizar una función de consolidación [84], se permite conservar los datos durante largos períodos de tiempo, dado que reduce gradualmente la precisión de los datos con mayor antigüedad a lo largo del tiempo. *RRDtool* presenta una granularidad para los datos de hasta un milisegundo.

Además, dicha herramienta cuenta con soporte para funciones de predicción [83], donde se permite realizar el pronóstico mediante el uso del método Holt-Winters, con la posibilidad de elegir cualquiera de las variaciones presentes en este método, ya sea aditiva o multiplicativa.

Sin embargo, dado que *RRDtool* no cuenta con una estructura cliente/servidor, carece de la posibilidad de escalabilidad y distribución, además de presentar un alto factor de carga de entrada/salida cuando se ejecutan actualizaciones. *RRDtool* permite una fácil integración con lenguajes como *Perl*, *Python*, *Ruby* y *Lua* [85].

2.5.4. OpenTSDB

OpenTSDB [90] es una base de datos de series temporales, distribuida y escalable, montada sobre *HBase* para almacenar los datos de series de tiempo. Utiliza *ZooKeeper* [5] para la coordinación entre nodos. Previo al inicio de *OpenTSDB*, se deben crear las tablas básicas en *HBase* para el funcionamiento de la base de datos. Para el ingreso de los datos, la estructura puede ser creada antes de la inserción, aunque *OpenTSDB* permite la generación automática de estas al detectar el ingreso de una nueva métrica a la base de datos. La granularidad de almacenamiento para los datos es de un segundo.

OpenTSDB tiene soporte con funciones de agregación como es el caso de *avg*, *count* y

sum [87], pero no cuenta con funciones de predicción propias, aunque dichas funciones están planeadas para su implementación en futuras versiones de la herramienta [89].

Para el uso de *OpenTSDB*, se requiere de *HBase* y *Gnuplot* como dependencias, además de *HBase* que requiere de *Zookeeper* para su completo funcionamiento. Se provee de interfaces de *HTTP* que permiten la generación de grafos mediante el uso de *Gnuplot*, y de *REST* para las diferentes funciones presentes en *OpenTSDB*. Además, cuenta con bibliotecas de cliente con soporte en *R*, *Erlang*, *Ruby*, *Python* y *Go* [88]. *OpenTSDB* soporta alta disponibilidad y escalabilidad mediante el uso de más de una instancia de la herramienta, mientras que al usar *Domain Name System* (DNS) Round Robin o herramientas como *HAProxy* [21] o *Varnish Cache* [93] permite realizar balanceo de cargas. Esta base de datos de series de tiempo es *open source* y no cuenta con soporte comercial.

2.5.5. Prometheus

Prometheus [64] es una solución de monitoreo a datos de series de tiempo basada en un modelo de extracción; de esta manera en vez de insertar datos, los solicita periódicamente, aunque es posible realizar una inserción de los datos mediante el uso de una segunda componente llamada *Pushgateway* [65]. Al utilizarla, se reemplaza la marca de tiempo de los datos a ingresar por una nueva dada por el momento en que se realiza la inserción. Debido a esto, esta herramienta no permite la inserción de datos históricos dado que no admite el campo de marca de tiempo al momento de ingresar los datos, siendo bastante limitado su uso. Este protocolo de texto utilizado por la segunda componente se denomina *Scraping*. La granularidad de almacenamiento para los datos es de un milisegundo.

Prometheus tiene soporte con funciones de agregación, dentro de las cuales se encuentran: *avg*, *count* y *sum* [62]. Proporciona un lenguaje de expresión funcional que permite realizar consultas de escritura y de agregación de datos de series temporales en tiempo real, además de hacer uso de sus funciones por sistemas externos a través de su *API HTTP*. Cuenta con la posibilidad de realizar un pronóstico de los datos mediante el uso del método de predicción Holt-Winters a través de la función de predicción *holt_winters*.

Prometheus permite escalabilidad y soporta alta disponibilidad mediante el uso de dos o más máquinas separadas que ejecuten servidores idénticos. Cuenta con bibliotecas de cliente con soporte en *Go*, *Java*, *Python* y *Ruby* [63]. Esta herramienta es *open source* y cuenta con un soporte comercial llamado *Robust Perception* [71], el cual apoya al cliente a medida que este utiliza *Prometheus* al profundizar sus aplicaciones como también en la infraestructura empleada.

2.5.6. Comparación entre Bases de Datos de Series de Tiempo

En esta sección se realiza una comparación de las bases de datos de series de tiempo mencionadas en la sección anterior. Los criterios a comparar se han agrupado en cuatro grupos, donde los tres primeros se basan en [3], mientras que el cuarto grupo se ha incorporado para contrastar con el interés en cada una de ellas. Estos cuatro grupos son:

1. Distribución y *Clustering*
2. Funciones, Almacenamiento a largo plazo y Granularidad
3. Extensibilidad y Soporte
4. Popularidad

El grupo 1 compara las siguientes características de Distribución y *Clustering*: alta disponibilidad, escalabilidad y balanceo de cargas. Alta disponibilidad permite compensar las fallas no esperadas en nodos y particiones de red; de esta manera, la alta disponibilidad es la calidad de un sistema para asegurar un alto nivel de rendimiento operativo durante un período de tiempo determinado. La escalabilidad es la capacidad de incrementar el almacenamiento o rendimiento al agregar más nodos sin perder calidad en los servicios ofrecidos. Finalmente, el balanceo de cargas es la posibilidad de distribuir las consultas entre los diferentes nodos de una base de datos de serie de tiempo, de modo que la carga de trabajo se distribuya de manera homogénea entre cada nodo y evite la sobrecarga en algún nodo en particular.

En el cuadro 2.2 se observa que tanto las tres características a comparar en este grupo se encuentran disponibles en tres bases de datos, mientras que *InfluxDB* a pesar de tenerlas se

encuentran restringidas a su versión empresarial. Por otro lado, solo *RRDtool* no permite llevar a cabo una distribución y *clustering* en su sistema.

TSDB	Alta Disponibilidad	Escalabilidad	Balanceo de Cargas
InfluxDB	Disponible con restricción	Disponible con restricción	Disponible con restricción
Graphite	Disponible	Disponible	Disponible
RRDtool	No Disponible	No Disponible	No Disponible
OpenTSDB	Disponible	Disponible	Disponible
Prometheus	Disponible	Disponible	Disponible

Cuadro 2.2: Comparación de criterios del grupo 1: Distribución y *Clustering* (elaboración propia).

El grupo 2 compara las características de disponibilidad de funciones de agregación y de predicción, almacenamiento a largo plazo y granularidad. Las funciones de agregación suelen ser utilizadas con frecuencia para el análisis de los datos; además, dado que el estudio se centra en el uso de consultas de tipo predictivas, se debe tener en consideración la disponibilidad de funciones de predicción en las bases de datos de series de tiempo. El almacenamiento a largo plazo, como su nombre lo indica, es la necesidad de guardar gran cantidad de datos, los que se pueden generar al trabajar con series temporales. La granularidad es una característica importante a considerar, pues determina la precisión con la que los datos son guardados, por lo que se debe tener en cuenta dicha característica para evitar la pérdida de información al momento de ingresar los datos, ya que éstos pueden tener una mayor granularidad de lo que soporta la herramienta.

En el cuadro 2.3 se observa que *OpenTSDB* y *Prometheus* no cumplen con la capacidad de almacenamiento a largo plazo. Las funciones de agregación son soportadas por todas las bases de datos de series en estudio, aunque hay diferencias entre ellas respecto a la cantidad de funciones de agregación presentes en la herramienta. Por otra parte, solo *OpenTSDB* no cuenta con el soporte a funciones de predicción; el resto de las herramientas permite realizar un pronóstico a través del método Holt-Winters. *InfluxDB* entrega una mayor precisión en el

almacenado de los datos a diferencia de *Graphite* que da una menor precisión, alcanzando una granularidad de hasta un segundo.

Las tres bases de datos de series de tiempo restantes alcanzan una granularidad de los datos almacenados de hasta un milisegundo.

TSDB	Funciones de agregación	Funciones de predicción	Almacenamiento a largo plazo	Granularidad
InfluxDB	Disponible	Disponible	Disponible	1 nanosegundo
Graphite	Disponible	Disponible	Disponible	1 segundo
RRDtool	Disponible	Disponible	Disponible	1 milisegundo
OpenTSDB	Disponible	No Disponible	No Disponible	1 milisegundo
Prometheus	Disponible	Disponible	No Disponible	1 milisegundo

Cuadro 2.3: Comparación de criterios del grupo 2: Funciones, Almacenamiento a largo plazo y Granularidad (elaboración propia).

El grupo 3 compara *API's*, interfaces, bibliotecas de cliente, complementos y soporte comercial[2]. Tal como se observa en el cuadro 2.4, solo *Prometheus* no cuenta con una interfaz de *CLI*, mientras que otras dos del grupo restante cuentan con una interfaz *HTTP*. Las cinco bases de datos de series de tiempo ofrecen bibliotecas de cliente de la *API* en diferentes lenguajes, en donde *Python*, *Ruby*, *Java* y *Go* son los lenguajes que mayormente se repiten entre las herramientas. Por otro lado, las cinco herramientas cuentan con el soporte de complementos. El soporte comercial se encuentra presente solo en *InfluxDB* y *Prometheus*.

[2]Engloba las *API's*, interfaces y bibliotecas de cliente oficiales.

TSDB	*API's* e Interfaces	Biblioteca de Cliente	Complementos	Soporte Comercial
InfluxDB	*CLI*, *Prometheus*, *CollectD*, *Graphite*, *OpenTSDB*, *UDP*, *HTTP(InfluxQL)*	*Go*, *Python*, *Ruby*, *Rails*, *Java*, *C#*, *R*, *Javascript*, *Scala*	Disponible	Disponible
Graphite	*CLI*, *Pickle*, *Render*, *UDP*	*Python*, *Javascript*, *Node.js*	Disponible	No Disponible
RRDtool	*CLI*	*Python*, *Lua*, *Perl*, *Ruby*	Disponible	No Disponible
OpenTSDB	*HTTP(REST + JSON, GUI)*, *CLI*	*Go*, *Ruby*, *R*, *Erlang*	Disponible	No Disponible
Prometheus	*Scraping*	*Go*, *Java*, *Python*, *Ruby*, *Scala*	Disponible	Disponible

Cuadro 2.4: Comparación de criterios del grupo 3: Extensibilidad y Soporte (elaboración propia).

El grupo 4 compara el puntaje de popularidad que *DB-Engines* [73] entrega a cada una de las bases de datos. El método de cálculo de este puntaje de popularidad involucra los siguientes parámetros [75]:

- **Número de menciones del sistema en la web:** número de resultados en las consultas realizadas en los motores de búsqueda de *Google*, *Yandex* y *Bing*.
- **Interés general en el sistema:** frecuencia de búsquedas en *Google Trends*.
- **Frecuencia de discusiones técnica sobre el sistema:** cantidad de preguntas relacionadas y cantidad de usuarios interesados en los sitios de *Stack Overflow* y *DBA Stack Exchange*.
- **Número de ofertas de trabajo en las que se menciona el sistema:** número de ofertas en los principales motores de búsqueda de empleo como *Indeed* y *Simply Hired*.

- **Número de perfiles en redes profesionales en los que se mencionaa el sistema:** se utilizan las redes profesionales más populares a nivel internacional, como *LinkedIn* y *Upwork*.
- **Relevancia en las redes sociales:** número de tweets de *Twitter*, en los que se menciona el sistema.

En el cuadro 2.5 se detallan los puntajes de popularidad propuestos por *DB-Engines* de cada motor de base de datos de series de tiempo en estudio, en el cual *InfluxDB* presenta la mayor puntuación por sobre el resto de las herramientas, seguida de *RRDtool*. Mientras que *Prometheus* es la que presenta el puntaje de popularidad más bajo entre los demás motores de bases de datos.

TSDB	**Puntaje de popularidad**		
	Junio 2018	**Mayo 2018**	**Junio 2017**
InfluxDB	11.33	11.00	8.20
RRDtool	2.67	2.68	3.02
Graphite	2.38	2.26	2.00
OpenTSDB	1.56	1.62	1.80
Prometheus	1.27	1.13	0.61

Cuadro 2.5: Comparación del criterio del grupo 4: Popularidad (elaboración propia).

2.6. Estudios similares

En esta sección, se presentan diferentes trabajos que se relacionan con el tema en estudio; además se describen distintos programas de prueba, también conocidos como *benchmarks*, que han sido utilizados en dichas investigaciones.

2.6.1. *Benchmarks*

Para evaluar el rendimiento de un sistema informático, es necesario del uso de alguna técnica o herramienta que permita la medición de sus componentes en un entorno controlable y definido. Esta técnica es denominada *benchmark* (o programa de prueba), la cual realiza la mencionada evaluación reproduciendo una carga de trabajo de manera genérica en dichos sistemas, permitiendo la comparación entre dos o más sistemas informáticos. Estos programas de prueba se agrupan en los denominados *benchmark suites*, los cuales se encargan de asociar diferentes programas con el objetivo de medir diferentes aspectos de los sistemas informáticos. Para que un *benchmark suite* cumpla con el objetivo principal de un programa de prueba, se estructuran algunos pasos a seguir para su diseño [48]:

1. **Determinar los objetivos del *benchmark*:** se revisa qué sistema informático se va a medir, especificando el tipo de carga de trabajo o entorno en el cual se va a evaluar dicho sistema.

2. **Analizar la carga de trabajo del(los) sistema(s):** se analiza la proporción, frecuencia o tamaño de las consultas en un servidor o en la ejecución de instrucciones, de manera que la carga de trabajo a utilizar sea genérica.

3. **Escoger los programas según los objetivos determinados:** los programas permiten la medición de diferentes parámetros en un sistema informático, donde se selecciona(n) el(los) programa(s) que se adapte(n) en mayor medida a los objetivos propuestos del *benchmark*, además de tener en cuenta el análisis de alternativas de las etapas anteriores en la elección de dicho(s) programa(s).

4. **Escoger las métricas o mediciones a tomar sobre el sistema:** se determinan las métricas según lo que se quiera medir en los sistemas informáticos, y en base a los análisis y elecciones realizadas en los pasos previos. Estas suelen ser de dos tipos: velocidad de una función del sistema informático y capacidad de la misma. Se le denomina métrica de tipo velocidad al tiempo que tarde en ejecutarse una consulta, programa o un conjunto de programas. Por otra parte, una métrica de tipo capacidad corresponde al número de usuarios que es capaz de soportar un sistema o bien, al número de

consultas que se puede ejecutar simultáneamente en un sistema informático.

5. **Se debe tener en cuenta los factores que influyan en el rendimiento:** se deben considerar todos los factores, tanto los que pueden variar durante el estudio (por ejemplo, versión del sistema operativo o cantidad de recursos disponibles) como también aquellos que pueden permanecer fijos. Cualquiera sea el caso, estos factores deben ser los mismos para todos los sistemas informáticos en estudio, para que tenga sentido realizar una comparación entre ellos.

6. **Realizar análisis de los resultados obtenidos:** es relevante resumir los resultados obtenidos de las mediciones, y sintetizarlas en algún esquema visual que permita centrar la atención en lo esencial de dichas evaluaciones. La presentación de mucha información sin ningún tipo de análisis no permite extraer buenas conclusiones del estudio realizado.

Existe una escasa variedad de *benchmarks* para datos de series de tiempo; de las pocas, se destacan principalmente tres de ellas: (1) *STAC-M3 Benchmark Suite*, (2) *YCSB-TS* y (3) *FinTime*, las que se detallan a continuación.

STAC-M3 Benchmark Suite corresponde a un *closed source benchmark*, que mide el rendimiento de las bases de datos de series de tiempo al ejecutar un análisis en dichos datos. Sin embargo, la especificación e información más detallada acerca de este *benchmark* se encuentra limitada solo para los miembros del consejo de dicha herramienta [77], por lo que no es apreciable su alcance.

YCSB-TS corresponde a una versión adaptada para las series de tiempo de *Yahoo! Cloud Server Benchmark* (*YCSB*) llamada *Yahoo! Cloud Server Benchmark for Time Series* [8], que permite usar funciones básicas de dominio de tiempo, además de contar con el soporte en el uso de marcas y rangos de tiempo. Este *benchmark* es usado en la herramienta *TSDB-Bench* [57] para medir el rendimiento de las bases de datos de series de tiempo. Para ello, cuenta con diferentes opciones de carga de trabajo; de esta manera permite conocer la latencia de las consultas y el consumo de espacio en diferentes escenarios, donde se incluye una configuración de *clusters* alterables.

FinTime es un *open source benchmark* para datos de series de tiempo con un enfoque financiero. Este *benchmark* usa principalmente dos modelos diferentes, los cuales se centran en el análisis de los siguientes parámetros: periodicidad de los datos, densidad de los datos, registro de actualizaciones, tipos de consultas, intervalo de tiempo entre consultas y número de usuarios simultáneos. Además, utiliza tres métricas para determinar el rendimiento de una base de datos en un entorno financiero realista con datos de series de tiempo [15]. No obstante, los modelos utilizados en este *benchmark* no coinciden con las estructuras de las bases de datos de series de tiempo modernas, estructura que posee tres componentes principales: métrica, valor y etiqueta. Por lo tanto, este *benchmark* posee un limitado soporte con las bases de datos de series de tiempo.

Estos *benchmarks* se relacionan principalmente en el enfoque en que realizan sus mediciones y análisis, otorgándole énfasis a las transacciones por sobre las consultas y estas que contengan funciones de agregación.

2.6.2. Trabajos realizados

En el año 2012, Tomasz Wiktor Wlodarczyk [81] compara cuatro soluciones para almacenar y procesar datos de series de tiempo, donde concluye que *OpenTSDB* [90] es la mejor opción si se necesita realizar un análisis avanzado, mientras que *TempoIQ* [82] es la mejor alternativa si se desea una solución de *software* como servicio, también conocido como *SaaS*.

Dos años después, Thomas Goldschmidt [80] realiza una comparación mediante los motores de bases de datos *OpenTSDB*, *KairosDB* [49] y *Databus* [53]. El objetivo de la investigación consiste en determinar si alguna de las bases de datos de series de tiempo pueden escalar linealmente, manejar cargas de trabajos de uso industrial, escalar cargas de trabajo independientes, tolerar el bloqueo de dos nodos y tener un rendimiento independiente de lectura y escritura de los datos. Luego de realizar las mediciones mediante el uso de dos cargas de trabajo diferentes y variando el tamaño de los *clusters*, concluye que solo el motor de base de datos *KairosDB* cumple todas sus hipótesis. Por otro lado, las pruebas de *OpenTSDB* son problemáticas, dado que los resultados obtenidos no son concluyentes.

Andreas Bader [3] en el año 2016, realiza una comparación de diferentes bases de datos de series de tiempo mediante el uso del programa *YCSB-TS*, la cual permite crear y medir las consultas de una carga de trabajo. Se utilizan dos métricas de rendimiento: (1) latencia en las consultas y (2) consumo de espacio; busca responder las interrogantes de qué tan rápido se ejecutan las diferentes consultas y como almacenan los datos para permitir un ahorro en el espacio utilizado. Además, se utilizan dos escenarios para el desarrollo de la comparación de las diferentes cargas de trabajo y bases de datos temporales en la investigación, en donde se consideran consultas de lectura, escritura y de agregación; siendo cada tipo de consulta a utilizar en ambos escenarios de igual proporción, excluyendo las consultas de escritura. Las diferentes bases de datos de series de tiempo presentes en el trabajo se dividen en cuatro grupos: (1) bases de datos con un requerimiento en sistemas de gestión de bases de datos *NoSQL*, (2) sin ningún requerimiento de sistema de gestión de bases de datos, (3) relacionales y (4) propietarias. De los resultados obtenidos concluye que en escenarios enfocados en el bajo uso de espacio o consultas: *read*, *scan*, *avg*, *sum* y *cnt*, *Druid* [14] es la mejor opción, mientras que si la prioridad se encuentra en las transacciones (escritura), se recomienda el uso de *OpenTSDB*, que presenta el menor tiempo promedio de latencia en éstas.

Capítulo 3

Propuesta de Diseño

En este capítulo se detalla el modelo de *benchmark* con el cual se lleva a cabo la comparación de consultas predictivas de las bases de datos de series de tiempo, donde se especifican los motores de bases de datos de series temporales a utilizar, el método predictivo seleccionado en las consultas, las características de los *sets* de datos, análisis y métodos de evaluación a realizar, y las métricas usadas para la precisión de los resultados.

3.1. Modelo a utilizar

En la sección 2.6 se identificaron diferentes estudios realizados que se relacionan con el *benchmark* de bases de datos, centrándose en el uso de bases de datos de series de tiempo. Estos aplican un modelo de *benchmarking* dirigido a la comparación de sistemas informáticos. Para el desarrollo de este trabajo se utiliza como base este modelo de *benchmarking*, dado el nivel de completitud que presenta al abarcar los puntos principales que se requieren en una comparación de sistemas, la cual será adaptada para una mejor sincronía con el objetivo del estudio, siendo incorporados componentes relacionados a la predicción de datos y una reestructuración de las etapas presentes en el modelo. En consecuencia, la propuesta se estructura en las siguientes fases: objetivos del *Benchmark*, carga de trabajo, selección de programas, métricas, factores que influyen en el rendimiento y análisis de los resultados.

Esta última fase será considerada en la implementación (ver capítulo 4).

3.2. Objetivos del *Benchmark*

El objetivo del *benchmark* consiste en comparar tres motores de bases de datos de series de tiempo al medir tanto la precisión como el rendimiento, centrándose en consultas predictivas, y así contar con un marco que permita establecer cuál es el más adecuado según sea la situación.

3.2.1. Bases de Datos de Series de Tiempo a comparar

En la sección 2.5 se introdujeron cinco de los principales motores de bases de datos de series de tiempo, los cuales son: *InfluxDB*, *RRDtool*, *Graphite*, *OpenTSDB* y *Prometheus*. Para el desarrollo de la memoria se decidió utilizar los tres primeros, dado el buen soporte que presentan con bibliotecas de cliente, interfaces y complementos, además de contar con un alto nivel de popularidad reflejado en el puntaje presentado por el ranking de *DB-Engines*. Por otra parte, *OpenTSDB* no cuenta con la implementación de funciones de predicción, por lo que a pesar de ser una base de datos de series de tiempo con un buen puntaje de popularidad, queda descartada del análisis. Mientras que *Prometheus*, debido a la política presente que evita insertar datos históricos, dificulta la carga de *sets* de datos externos, por lo cual no será considerada en el estudio.

3.2.2. Método Predictivo a utilizar

En la sección 2.3 se especificaron diferentes métodos de predicción, los cuales son: modelo ARIMA, Suavizamiento Exponencial Simple, Método de Holt y Método de Holt-Winters. De los cuales, ARIMA y Holt-Winters corresponden a métodos más completos que los otros descritos, siendo los que más se consideran al momento de realizar estudios predictivos, por su sencilla implementación y fácil uso de sus parámetros, además de presentar cálculos de

bajo costo de cómputo. En este trabajo se utilizará el método Holt-Winters dado el soporte que presenta con las funciones predictivas de los motores de bases de datos de series de tiempo en estudio, por sobre el modelo ARIMA.

3.3. Carga de trabajo

En la fase anterior se ha seleccionado el método Holt-Winters como método predictivo, el cual se centra en pronósticos a corto plazo. Dado esto, la carga de trabajo se seccionará en tres escenarios:

- **Predicción corta (Escenario 1):** el primer escenario realiza pronósticos a partir de un día hasta un máximo de 6 días, con una granularidad de predicción de un día. Es decir, los posibles pronósticos son: 1, 2, 3, 4, 5 y 6 días.
- **Predicción media (Escenario 2):** el pronóstico del segundo escenario considera desde 7 días hasta un límite de 28 días, siendo 7 dias la granularidad de predicción en este escenario. De esta manera, los posibles pronósticos son: 7, 14, 21 y 28 días.
- **Predicción extensa (Escenario 3):** el tercer escenario realiza predicciones desde los 35 días hasta llegar a un límite máximo de 56 días. Es así como los posibles pronósticos son: 35, 42, 49 y 56 días.

Cada consulta predictiva para los posibles pronósticos en los escenarios antes mencionados se presentarán en iguales cantidades; de esta manera, se le otorga la misma importancia a cada consulta realizada en los diferentes escenarios, lo cual permite que la carga de trabajo a utilizar sea genérica.

3.3.1. *Sets* de Datos a utilizar

En la selección de los *sets* de datos a considerar en este trabajo, se busca abarcar las diferentes situaciones que se presentan en la práctica. De esta manera, se utilizarán tres *sets* de datos:

(1) un *set* de datos pequeño, el cual cuenta con una baja cantidad de registros y atributos, (2) un *set* de datos con presencia de muchos atributos y (3) un *set* de datos que contiene muchos registros. Por otra parte, la opción de un *set* de datos que contenga gran cantidad tanto de registros como atributos no se considera en el estudio, debido a que la manipulación de dicho conjunto de datos está fuera del alcance de los recursos disponibles. Los *sets* de datos a utilizar en esta memoria se detallan a continuación:

- ***Set* de datos pequeño (*Dataset 1*):** conjunto de datos denominado *Air Quality Data Set* [94], el cual contiene respuestas de un dispositivo multisensor de gas desplegado en el campo de una ciudad italiana, en un área significativamente contaminada. Los datos fueron registrados desde el mes de marzo del 2004 hasta febrero de 2005, y representan concentraciones promediadas por hora de diferentes compuestos químicos. Este *dataset* cuenta con 9.358 registros y 15 atributos.
- ***Set* de datos con muchos atributos (*Dataset 2*):** conjunto de datos denominado *Electricity Load Diagrams Data Set* [91], el cual contiene el consumo de electricidad de diferentes puntos de cada cliente. El *dataset* cuenta con 140.256 registros y 370 atributos, donde cada registro es un punto que se analiza el consumo de la electricidad, y cada atributo representa un cliente. Además, los valores de cada punto en estudio fueron registraron cada 15 minutos.
- ***Set* de datos con muchos registros (*Dataset 3*):** conjunto de datos denominado *Individual Household Electric Power Consumption Data Set* [16], el cual contiene mediciones del consumo de energía eléctrica en un hogar, registrando el muestreo de los datos cada 1 minuto desde el mes de diciembre del 2006 hasta el mes de noviembre del 2010. El *set* de datos cuenta con 2.075.259 registros y 9 atributos.

3.3.2. Método de Entrenamiento

El método de entrenamiento a utilizar es el llamado *K-fold Cross Validation* [67], considerando 5 *folds* estratificados para las diferentes pruebas a realizar. Dado que las series de

tiempo son estrictas en la cronología del tiempo y no permiten ir modificando las ubicaciones de los datos ya registrados en ellas, se maneja una variante de este método para series de tiempo [70], la cual, tal y como se presenta en la figura 3.1, permite que no se perturbe la cronología de los datos, pues cada iteración va sumando el *set* de entrenamiento y de prueba de la iteración actual y el resultado es usado como el *set* de entrenamiento de la iteración siguiente. Esto da lugar a que se calculen predicciones variando la cantidad de datos históricos con que se cuenta, y de esta manera, permite que el resultado a obtener sea representativo de diferentes situaciones referente al volumen de datos históricos a utilizar y no sea solo el resultado de un caso aislado. Es por esto que se ha optado por el uso de *K-fold Cross Validation* por sobre otros métodos como lo es, por ejemplo, el uso de un 70 % del *set* de datos para entrenamiento (datos históricos) y el 30 % restante como *set* de prueba.

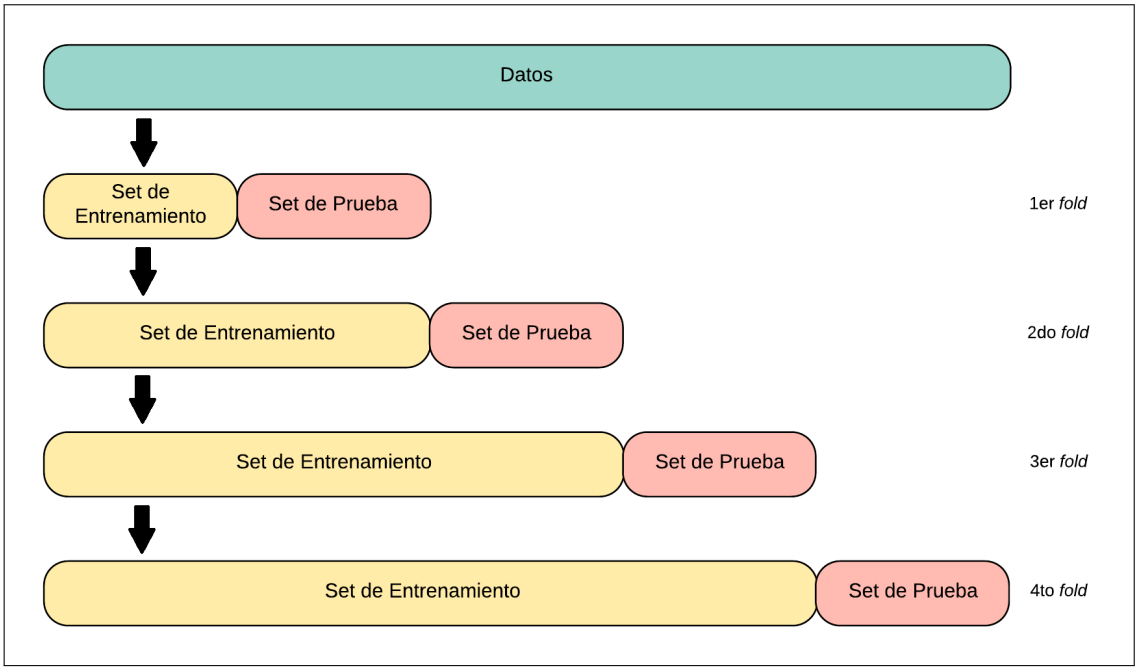


Figura 3.1: Diagrama de ejemplo de representación del método *K-fold Cross Validation* para series de tiempo utilizando 4 *folds* como estratificación (elaboración propia).

3.4. Selección de Programas

En la sección 2.6.1 se detallaron las herramientas utilizadas en el desarrollo de un *benchmark*, incorporando en la mayoría de ellas el estudio de las bases de datos de series de tiempo. Sin embargo, la comparación que estas realizan se focalizan principalmente en las funciones de selección y agregación. Dado que las funciones de predicción requieren de otro enfoque para su análisis, puesto que no solo se debe considerar el rendimiento sino que también la precisión de los resultados, es que para el estudio se decide implementar un *script* que permita realizar una comparación que se adecúe de mejor manera al objetivo de trabajo.

El *script* generado permite ejecutar diferentes consultas predictivas, las cuales serán comparadas según sea las distintas métricas a utilizar en el estudio. Para esto, se requiere que sea capaz de realizar las conexiones necesarias para que las consultas en cada motor de base de datos puedan ser ejecutadas de manera remota. Luego de ejecutar cada consulta predictiva, esta es contrastada con los datos originales presentes en el *set* de datos empleado, para medir la precisión de los resultados.

En cada *test* se varía tanto el escenario como el *dataset* utilizado, de esta manera se aplican 9 *tests* diferentes en este trabajo. Por otra parte, cada *set* de consultas considera un total de 12 consultas predictivas; esto, debido a que corresponde al minimo común múltiplo entre los diferentes casos posibles en cada uno de los escenarios, permitiendo así que cada caso cuente con la misma cantidad de consultas. No se considera una cantidad mayor de consultas en el *set*, ya que estas cuentan con una estructura definida, y por ende, se limita el rango de posibilidades en que se puede presentar una consulta predictiva. A su vez, cada *set* de consultas se ejecuta 5 veces; esta repetición del *set* permite contar con un respaldo de los resultados obtenidos, asegurando así que lo ejecutado no es parte de un caso aislado.

3.5. Métricas

En la sección 3.2 se detalló lo que se quiere medir en los motores de bases de datos de series de tiempo; las métricas a utilizar en las mediciones se dividen en dos tipos: de precisión y

de rendimiento. El objetivo del primer tipo de métricas consiste en determinar qué motor de base de datos presenta una predicción más cercana al comportamiento ocurrido en la práctica. Para ello, en el estado del arte se identificaron diferentes métricas de error, dentro de las cuales se decidió por utilizar tanto el MAPE como el MASE, puesto que son los errores que presentan un valor significativo en una comparación con *set* de datos que difieren en su granularidad de almacenamiento de sus registros. Por otro lado, el MAE y el RMSE pierden validez en la comparación de set de datos con diferentes intervalos de separación de tiempo de sus registros, por lo que estas dos últimas métricas no son consideradas en el estudio.

Por otra parte, para las métricas de rendimiento se utilizan métricas de velocidad, como el tiempo de ejecución de las consultas, puesto que el tiempo es un factor importante en el costo de una consulta. Además, se emplean métricas de aprovechamiento de la máquina, como porcentaje de uso de la CPU y uso de memoria, dado que son estas la que determinan si existe algún requerimiento minimo en *hardware* para utilizar los motores de bases de datos de series de tiempo.

3.6. Factores que influyen en el rendimiento

En esta sección se especifican los diferentes factores que son utilizados en el desarrollo de este trabajo, los cuales pueden condicionar el rendimiento presente en el experimento.

3.6.1. *Hardware* y Herramientas a utilizar

Para la implementación del *benchmark* se utilizan dos equipos, donde sus características se encuentran en el cuadro 3.1.

	Sistema Operativo	CPU	RAM
Equipo A	Windows 10 64 bits	Intel Core i7 3.60[GHz]	16 GB
Equipo B	Windows 10 64 bits	Intel Core i7 3.60[GHz]	8 GB

Cuadro 3.1: Especificaciones de los equipos utilizados para la implementación del *benchmark* (elaboración propia).

En el equipo A se montan tres máquinas virtuales, las cuales corresponden a cada uno de los motores de bases de datos de series de tiempo en estudio. Estas son: *InfluxDB*, *Graphite* y *RRDtool*, las cuales tienen las siguientes especificaciones:

- Sistema Operativo: CentOS 7 64 bits
- CPU: 2 vCPU
- RAM: 4 GB

Por otra parte, el equipo B cuenta con dos máquinas virtuales, la principal y la de apoyo. La primera es la encargada de ejecutar el *script* correspondiente para el desarrollo del *benchmark*, mientras que la segunda cumple un rol de apoyo ante ciertos datos que se requieran para realizar la comparación. Las especificaciones de cada máquina se encuentran en el cuadro 3.2.

	Sistema Operativo	CPU	RAM
Máquina Principal	CentOS 7 64 bits	2 vCPU	3 GB
Máquina de Apoyo	CentOS 7 64 bits	2 vCPU	4 GB

Cuadro 3.2: Especificaciones de las máquinas virtuales montadas en el equipo B (elaboración propia).

Para la implementación del *script* se utiliza Python en su versión 2.7, debido a las bibliotecas que este lenguaje de programación dispone, siendo un apoyo en el desarrollo del experimento.

3.6.2. Arquitectura del Entorno

En la figura 3.2 se presenta la arquitectura del entorno utilizado en el trabajo, la cual cuenta con tres máquinas virtuales correspondientes a cada uno de los motores de bases de datos de series de tiempo (*InfluxDB*, *Graphite* y *RRDtool*). Además, se encuentra la máquina principal la cual se conecta a las tres mencionadas anteriormente para poder ejecutar las diferentes consultas de los *tests* a realizar. A su vez, esta utiliza la máquina de apoyo como respaldo en el desarrollo del proceso de *benchmark*.

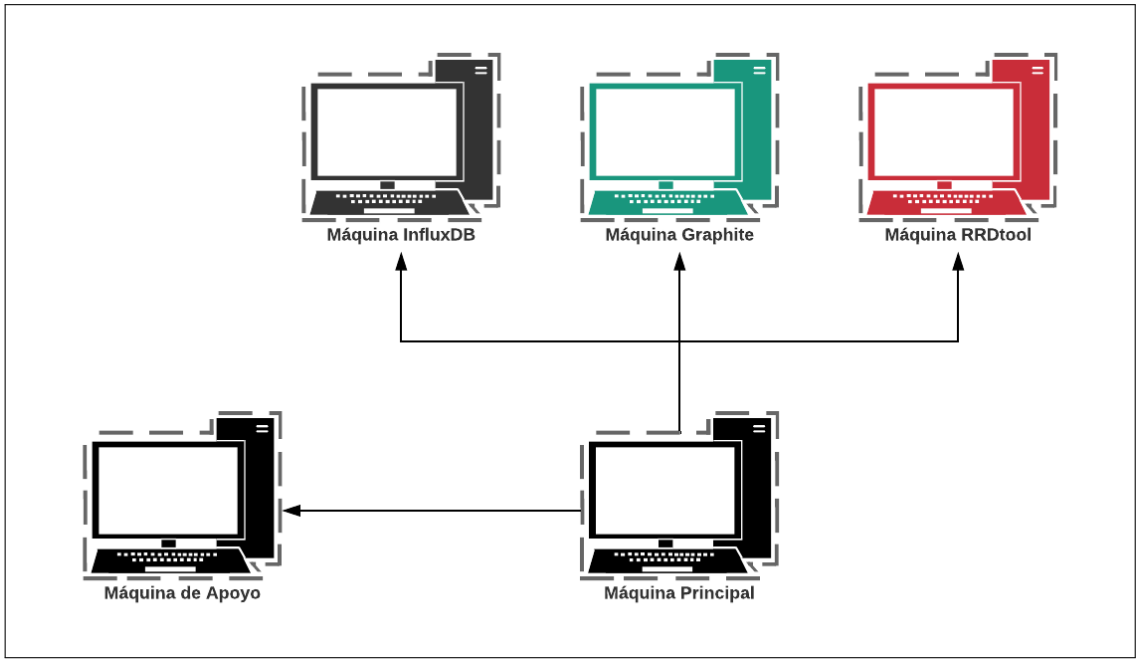


Figura 3.2: Arquitectura del entorno utilizado en el desarrollo de la experimentación (elaboración propia).

Capítulo 4

Implementación y Pruebas

En este capítulo se detallan los pasos generales para la construcción del entorno de trabajo en cada motor de base de datos, así como también las funciones principales en la implementación del *script* y el procedimiento que se utiliza en la ejecución de las pruebas del *benchmark*.

4.1. Construcción del Entorno

En el desarrollo del experimento, para la construcción del entorno de trabajo se realiza la virtualización de cada máquina mediante el uso de *VirtualBox* [58] en su versión 5.2, que corresponde a un *software open source* disponible para uso público. En la presente sección se detallan los pasos generales de la implementación de estas, y el trabajo previo realizado para la carga de los *sets* de datos a los diferentes motores de base de datos.

4.1.1. Máquina Principal

Corresponde a la máquina virtual donde se localiza el *script* encargado para la experimentación, y es la que se conecta con el resto para llevar a cabo el *benchmarking*. Por otra parte, y dado que esta tiene conexión con las demás, se instala *Grafana* en su versión 5.1 [17],

herramienta *open source* para el monitoreo y análisis de datos, especializada en el análisis de series de tiempo.

4.1.2. Máquina InfluxDB

En la máquina *InfluxDB* se encuentra el entorno de trabajo para el funcionamiento del motor de base de datos *InfluxDB* [38] en una arquitectura con un solo nodo, versión 1.4.2. Por otra parte, se requiere de la configuración de una herramienta para la medición del rendimiento. Por ello, si bien *influxdata* cuenta con una herramienta que complementa la base de datos y que cumple con lo requerido, se decide utilizar un *software* genérico para todos los motores de base de datos en estudio con el objetivo de que las condiciones de cada base de datos sean iguales. Es por esto que se instala la herramienta *collectd* [11] en su versión 5.8.0, disponible para uso público, para la medición del rendimiento, puesto que cuenta con variados *plugins* para las bases de datos en estudio, lo que permite una mejor manipulación de los datos almacenados.

Antes de realizar un pronóstico mediante el uso de consultas predictivas, hay que tener en cuenta la configuración que se utilizará al momento de ejecutar dicha consulta. Es así como en la experimentación se hace uso de la función *holt_winters*, la cual cuenta con los siguientes parámetros:

- **Función de datos**: tendencia de los datos históricos a utilizar en la predicción. Esta función está dada por aquella obtenida tras realizar una consulta en la base de datos que coincida con los *peaks* presentes en los datos históricos.
- **N**: cantidad de valores a predecir.
- **S**: patrón estacional, delimitando su duración mediante un intervalo de tiempo.

Para seleccionar el valor del primer parámetro, la documentación de *InfluxDB* recomienda obtenerlo mediante un método de "prueba y error" mientras se ejecuta la consulta hasta encontrar la que represente la tendencia de los datos. Sin embargo, dado que en un proceso de *benchmarking* se recomienda la elección del valor de una manera automatizada, debido a

la cantidad de ejecuciones que esta se realiza (siendo no viable obtener dicho valor de forma manual), se ha optado por calcular dicho parámetro de la siguiente manera:

- Se obtienen los *peaks* estimados de los datos históricos.
- Se calcula la media de la diferencia en el intervalo de tiempo entre un *peak* y el *peak* siguiente.
- La función de datos es agrupada por el tiempo utilizando el valor de la media calculada en el paso anterior.

Por otra parte, el parámetro *N* es calculado mediante la división de la cantidad de tiempo que se desea predecir y el intervalo de tiempo presente entre cada medición de los datos históricos. Mientras que el parámetro *S* es obtenido mediante el uso de la biblioteca *seasonal* [96], la cual estima periodicidad en series de tiempo mediante el uso de una técnica de promediado de periodograma[1] denominado método Welch [61].

4.1.3. Máquina Graphite

En la máquina *Graphite* se encuentra el entorno de trabajo para el funcionamiento del motor de base de datos *Graphite* [20] con una arquitectura de un solo nodo, usando la versión 1.1.1. La medición del rendimiento se realiza mediante el uso de la misma herramienta empleada en la máquina descrita en la sección 4.1.2, es decir, *collectd* en su version 5.8.0.

Previo a realizar un pronóstico mediante el uso de consultas predictivas, hay que tener en cuenta la configuración que se utilizará al momento de ejecutar dicha consulta. En la experimentación se hace uso de la función *holtWintersForecast* para obtener un pronóstico; sin embargo se debe tener en cuenta sus parámetros al momento de ejecutar una consulta. La función *holtWintersForecast* cuenta con los siguientes parámetros:

- ***seriesList***: nombre de la propiedad del *dataset* sobre el cual se realiza el pronóstico.

[1] El periodograma es un estimador espectral que sirve para detectar estacionalidad en una serie y determinar su período.

- ***bootstrapInterval***: intervalo de tiempo que es utilizado como datos históricos para realizar el pronóstico.

La selección de ambos parámetros está dada por los valores presentes en cada escenario de prueba, donde arbitrariamente se elige una métrica para calcular el pronóstico utilizando a su vez un intervalo de datos históricos definido en el experimento.

4.1.4. Máquina RRDtool

En la máquina *RRDtool* se encuentra el entorno de trabajo para el funcionamiento del motor de base de datos *RRDtool* [86] usando una arquitectura de un solo nodo, en su versión 1.7.0. Para la medición del rendimiento se instala, nuevamente, *collectd* en su version 5.8.0. Para permitir que la herramienta *Grafana* pueda leer los datos recopilados, guardados en archivos RRD, se utiliza la biblioteca *Grafana RRD Server* [22], la cual consiste en un servidor HTTP que otorga la facilidad de responder antes las consultas realizadas desde *Grafana*.

Por otra parte, para realizar una predicción se hace uso de la función *hwpredict* la cual, en conjunto con la función *seasonal*, permite obtener un pronóstico estacional. Para esto se cuenta con los siguientes parámetros:

- ***rows***: longitud del *round robin archive* (RRA), archivo donde se almacenan los valores de los datos o estadísticas de ella.
- ***alpha***: suavizado del coeficiente de intercepción en el algoritmo de pronóstico de Holt-Winters.
- ***beta***: suavizado del coeficiente de pendiente en el algoritmo de pronóstico de Holt-Winters.
- ***gamma***: suavizado de los coeficientes estacionales en el algoritmo de pronóstico de Holt-Winters.
- ***seasonal period***: cantidad de datos presentes en un ciclo estacional.

La selección del parámetro *rows* está dada por la división entre la diferencia de tiempo entre el primer valor de los datos históricos y el último valor a predecir, y el intervalo de tiempo que separa un punto de dato del siguiente. Mientras que los parámetros de suavizado se calcularon mediante el uso del método heurístico de "prueba y error" en la evaluación del modelo Holt-Winters aditivo, donde se optimiza el RMSE, para lo cual se utiliza la biblioteca *scipy* para realizar dicha optimización. Finalmente, el parámetro *seasonal period* se obtiene mediante el uso de la biblioteca *seasonal*, estimando de manera similar al parámetro *S* de la función de predicción vista en la sección 4.1.2. Por otro lado, dado que en *RRDtool* se almacenan los datos en archivos RRD, estos deben ser creados previamente al cargado de los datos, lo que significa que la configuración requerida para realizar un pronóstico se especifica con anterioridad a la experimentación, sin dar lugar a que dicha configuración sea modificada. Es por esto que se emplea la función *rrd_hwreapply* disponible en el proyecto *RRDman* [78], la cual permite la manipulación de los parámetros de predicción del algoritmo Holt-Winters en un archivo RRD.

4.1.5. Máquina de Apoyo

La máquina de apoyo corresponde a la máquina virtual que sirve como respaldo ante datos que se requieran durante la experimentación. Es así como esta máquina contiene una copia de los tres *sets* de datos a emplear para el desarrollo de este trabajo. Además los valores calculados, ya sea de medidas de error o tiempo de ejecución, son almacenados en la base de datos presente en esta máquina. Debido a la facilidad en su instalación y configuración, se utiliza el motor de base de datos *InfluxDB* en su versión 1.4.2.

4.1.6. Manejo de los datos y Poblamiento de las bases de datos

El poblamiento de las bases de datos se realiza mediante la transformación del formato de los *sets* de datos al requerido en cada motor de base de datos, además de la selección de los atributos en cada *dataset*, filtrando aquellos que pueden generar incertidumbre durante la experimentación siendo contraproducente su uso en el *benchmark*, o bien ciertos atributos

que no son necesarios de incluir para el desarrollo de este trabajo. En el caso de *InfluxDB*, la carga de los datos se hace con el comando *import* presente en el *CLI*. Por otro lado, el cargado de los datos en *Graphite* es por medio del *plaintext protocol*, mientras que para el motor *RRDtool* se emplea la función *rrdtool update*. Las transformaciones de los datos en cada *dataset* son detalladas a continuación:

- **Dataset 1**: se modifican los atributos relacionados con la fecha y hora de los registros, unificando todos en un solo atributo de tipo *timestamp*, debido a que es este el formato que se utiliza en las bases de datos de series de tiempo. Además, se elimina del *dataset* el atributo "*NMHC(GT)*", puesto que la mayor parte de los registros se encuentran nulos, por lo que no aporta información al estudio.

- **Dataset 2**: cuenta con un atributo donde incluye la fecha y hora de los registros, el que se modifica al tipo *timestamp*. Por otra parte, puesto que los primeros 24 registros del *dataset* se encuentran nulos en la mayoria de los atributos, se opta por eliminarlos. Debido a que cuenta con una gran cantidad de atributos que no serán abarcados en su totalidad en la experimentación, se decide utilizar 75 atributos del total de ellos para el estudio; descartar parte de los atributos no genera incertidumbre puesto que solo se disminuye el umbral de clientes, considerando que cada atributo hace relación al consumo de un cliente.

- **Dataset 3**: se modifican los atributos relacionados con la fecha y hora de los registros, unificando todos en un solo atributo de tipo *timestamp*. Por otra parte, puesto que tres de los atributos existentes ("*Sub_metering_1*", "*Sub_metering_2*" y "*Sub_metering_3*"), cuentan con valores muy repetitivos y que su variación es en períodos largos, se opta por eliminarlos, debido a que generan incertidumbre y ruido al momento de realizar un pronóstico en dichos atributos.

Uno de los requisitos al momento de utilizar series de tiempo es que no debe existir ningún valor nulo en la cronología de los datos, y dado que los *sets* de datos presentan valores nulos en sus atributos, se decide usar el método de interpolación para estimar los espacios de tiempo sin datos; para ello se emplea la función *interpolate* presente en la biblioteca *pandas* [95].

4.2. Codificación del *script*

Para el desarrollo del *benchmark* se decide el uso de un *script* propio que permita generar un escenario que se adapte de mejor manera al enfoque de este trabajo. Para su implementación este se ha dividido en tres partes:

- **Configuración inicial y conexión a la base de datos**: en primera instancia, se definen los elementos generales a utilizar en el *script*, vale decir, se configuran el *set* de datos a usar en la experimentación, el escenario a considerar para realizar la prueba, la cantidad de consultas a generar para la comparación y el número de repeticiones que se ejecutará el *set* de consultas. Además se inicializan las conexiones a las bases de datos, que son utilizadas para enviar las diferentes consultas durante el experimento. En el caso de *InfluxDB* se realiza la conexión mediante el uso de la biblioteca *InfluxDB-Python* [33], la que permite la interacción con dicho motor de base de datos. Por otra parte, la aplicación web que dispone *Graphite* provee la API *Render URL*, que posibilita la recuperación de los datos solicitados. Es por esto que la conexión a este motor consiste en generar la *URL* para acceder a los datos presentes en dicha herramienta, donde cada solicitud se ejecuta utilizando la función *get* correspondiente a la biblioteca *Requests* [50], la que envía la solicitud *HTTP* y recupera la información del resultado obtenido. Finalmente, puesto que *RRDtool* presenta un enfoque local en el manejo y ejecución de consultas, y dado que no presenta aplicaciones que ejecuten dichas solicitudes de manera remota, se decide realizar la conexión a dicha base de datos a través de una conexión *SSH* mediante el uso de la función *pxssh*, disponible en la biblioteca *pexpect* [55].

- **Generación de consultas predictivas**: una vez definida la configuración inicial, se ejecuta el proceso de la generación del *set* de consultas a ser utilizado para la comparación de los motores de base de datos. Este proceso usa la función *query_generator* (desarrollada en esta memoria), la cual se encarga de seleccionar los parámetros requeridos para la construcción de una consulta predictiva, considerando las condiciones establecidas en cada escenario de prueba. Esta función se apoya en otras como *query_mining* y *new_query* (también desarrolladas en esta memoria) para originar el *set* de consultas.

La primera de ellas se encarga de obtener consultas representativas del *dataset* para ser agregadas al *set* de consultas; para esto se utiliza el *p-value* asociada a la distribución *F* de Fisher presente en el *test* ANOVA[2] para determinar rangos del *set* de datos que sean característicos al *dataset*. De esta manera, se va iterando sobre el rango de tiempo de una métrica del *set* de datos, y mediante el uso de un factor diferencial determina la separación presente entre cada iteración. Posteriormente, al conocer todos los parámetros y valores requeridos para realizar una consulta predictiva, a través de la función *new_query* se genera la consulta predictiva para los diferentes motores de bases de datos de series de tiempo en estudio. Finalmente, este proceso retorna los *sets* de consultas correspondientes a la estructura de cada motor de base de datos.

- **Ejecución de iteraciones**: luego de que el *set* de consultas se encuentra ya generado se da inicio a la experimentación. Para esto se comienza a iterar cada consulta presente en el *set* y se procede a su ejecución en cada motor de base de datos; luego, utilizando los resultados obtenidos y los datos reales presentes en el *dataset*, se calculan las medidas de error MAPE y MASE. Seguido de esto, se almacenan los datos obtenidos en la base de datos presente en la máquina de apoyo y se lleva a cabo la siguiente iteración. Una vez que el *set* de consultas ha sido ejecutado totalmente, se cambia aleatoriamente el orden de las consultas que se encuentran en el *set*, a modo de evitar que el orden en que estas fueron ejecutadas permita otorgar ventaja en algunas consultas, como lo es por ejemplo, que cierta información se encuentre disponible en la memoria caché. Posteriormente a esto, se vuelve a iterar el *set* de consultas repitiendo el procedimiento hasta completar el número de veces a ejecutar (parámetro configurado anteriormente).

4.3. *Benchmark*

En esta sección se presentan los resultados obtenidos al realizar el *benchmark*. Para ello, se divide en tres subsecciones las cuales detallan los resultados de los diferentes escenarios en cada *dataset* utilizado en la evaluación de dicho *benchmark*.

[2]ANOVA corresponde a un análisis que permite probar la hipótesis de que si las medias de dos o más poblaciones son iguales [54].

4.3.1. *Dataset* 1

El desarrollo del *benchmark* para el *dataset* 1 se inicia con la evaluación del primer escenario, donde los resultados que se obtienen son divididos en dos partes. La primera da a conocer el rendimiento durante la ejecución, como muestra la figura 4.1, el cual deja en evidencia el uso de la CPU de cada motor de base de datos, en el que *InfluxDB* muestra un mayor porcentaje de uso de la CPU alcanzando los 14,64 %, mientras que *Graphite* emplea solo un 2,86 %. Por otra parte, la figura 4.2 grafica la cantidad de memoria RAM que se utiliza durante el *test*, donde se observa que la memoria se mantiene constante sin presentar alguna alza en ella, siendo el promedio de la memoria de 778,76 MB, 737,05 MB y 749,61 MB para los motores *InfluxDB*, *Graphite* y *RRDtool*, respectivamente. A su vez, el tiempo de ejecución de cada iteración en el *benchmark* se encuentra en la figura 4.3; *InfluxDB* presenta tiempos irregulares y es la herramienta que alcanza el mayor tiempo de ejecución promedio, siendo este de 126,79 ms. En cambio, *Graphite* es el que obtiene el menor tiempo promedio con un valor de 18,57 ms.

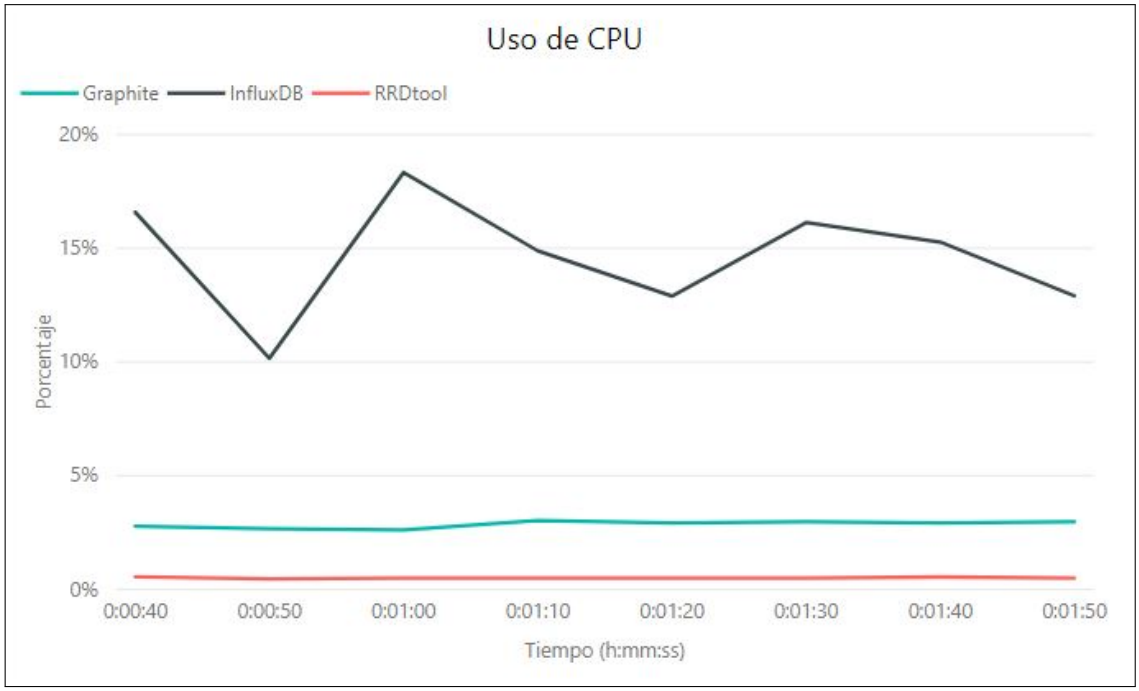


Figura 4.1: Uso de la CPU en función al tiempo de ejecución del primer escenario presente en el *dataset* 1 (elaboración propia).

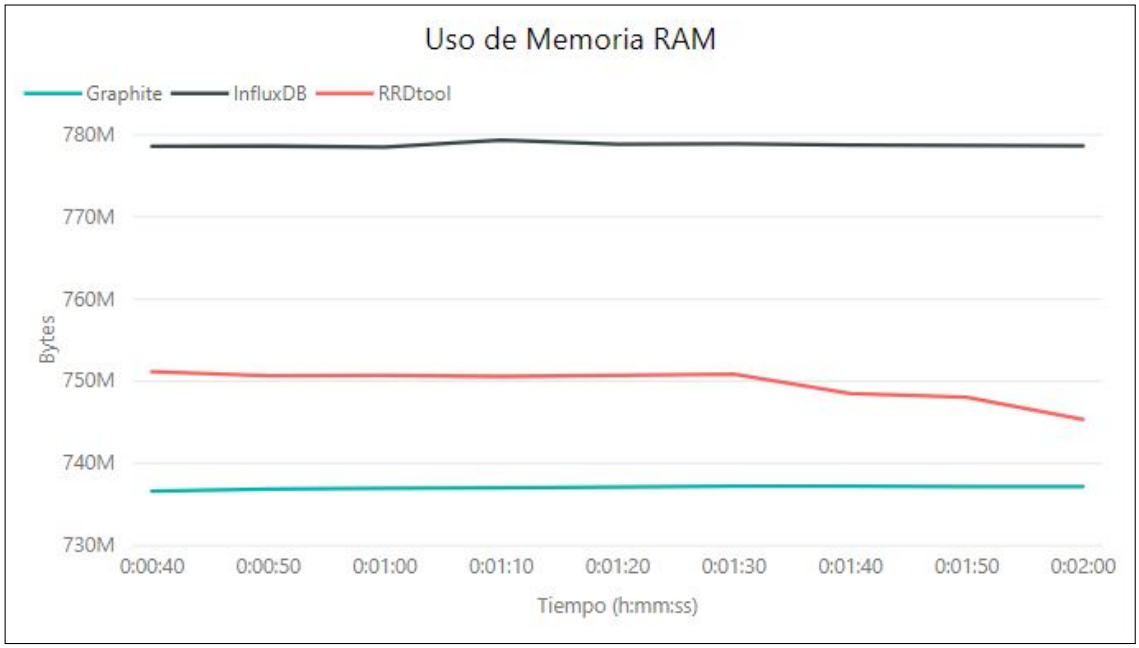


Figura 4.2: Uso de memoria RAM en función al tiempo de ejecución del primer escenario presente en el *dataset* 1 (elaboración propia).

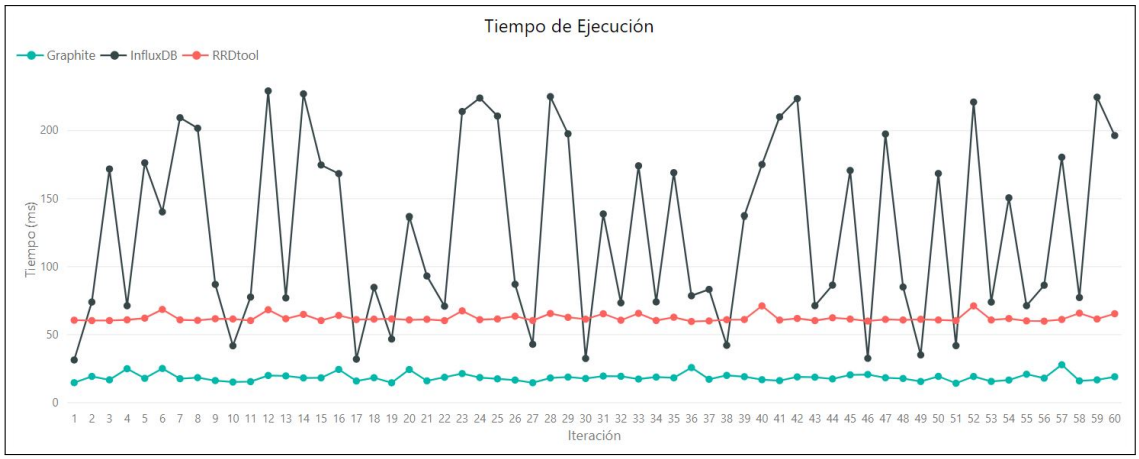


Figura 4.3: Tiempo de ejecución de cada iteración correspondiente al primer escenario del *dataset* 1 (elaboración propia).

Por otro lado, la segunda parte de los resultados da a conocer la precisión de las predicciones mediante las medidas de error MAPE y MASE, presentes en las figuras 4.4 y 4.5, los cuales dan a conocer el promedio de dichas medidas durante el desarrollo del *test*.

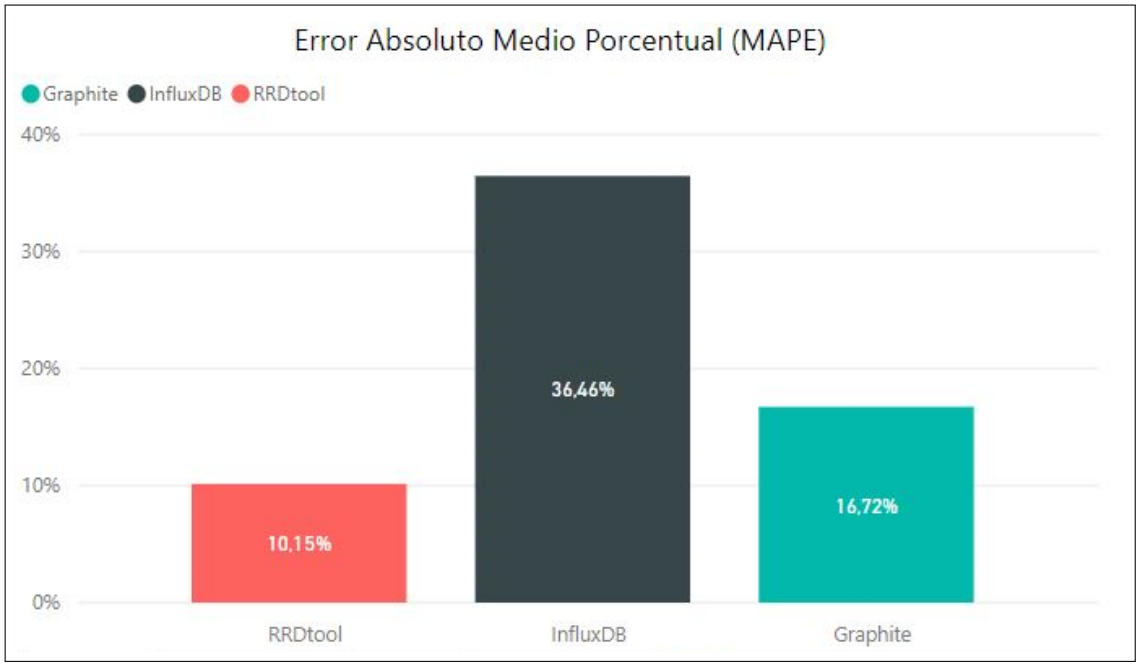


Figura 4.4: Promedio del MAPE correspondiente al primer escenario del *dataset* 1 (elaboración propia).

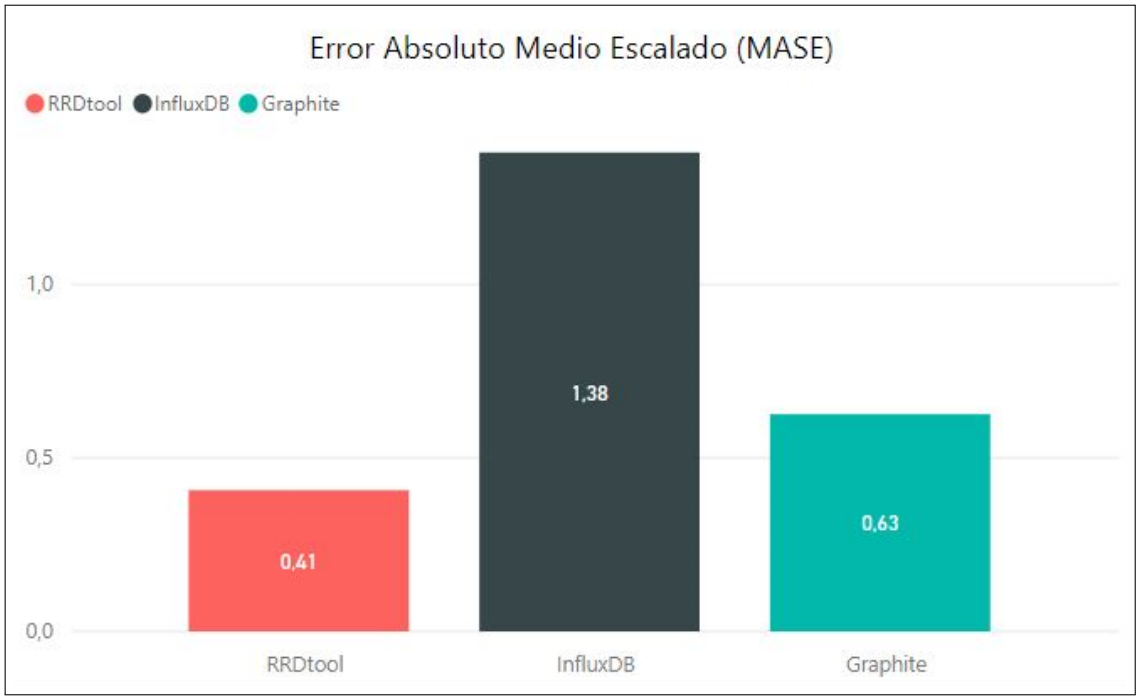


Figura 4.5: Promedio del MASE correspondiente al primer escenario del *dataset* 1 (elaboración propia).

Resumen de resultados en el primer escenario del *dataset* 1

TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	14,64 %	778,76 MB	126,79 ms	36,46 %	1,38
Graphite	2,86 %	**737,05 MB**	**18,57 ms**	16,72 %	0,63
RRDtool	**0,51 %**	749,61 MB	62,23 ms	**10,15 %**	**0,41**

Cuadro 4.1: Resultados promedios de las métricas medidas al ejecutar el primer escenario del *dataset* 1 (elaboración propia).

Posteriormente, se realizó el desarrollo del *test* para el segundo escenario siguiendo el mismo procedimiento ejecutado en el anterior. De esta manera, las figuras 4.6, 4.7 y 4.8 dan a conocer el rendimiento de cada motor de base de datos durante la evaluación de este escenario. Es así como se presentan los usos promedios de CPU, siendo 30,48 %, 1,69 % y 0,29 % correspondientes a *InfluxDB*, *Graphite* y *RRDtool*, respectivamente. A su vez, en los tiempos de ejecución, *Graphite* y *RRDtool* presentan tiempos estables que no superan los 80 ms; por el contrario, *InfluxDB* muestra una alza en el tiempo de algunas de sus iteraciones, alcanzando un tiempo de ejecución promedio de 616,42 ms.

A su vez, las figuras 4.9 y 4.10 dan cuenta de la precisión obtenida en este *test*; *RRDtool* resulta ser el motor con mejor precisión, obteniendo un 9,42 % en el MAPE y un 0,28 para el MASE. Este último indica que dicha herramienta genera el mejor modelo predictivo respecto a los demás motores de base de datos.

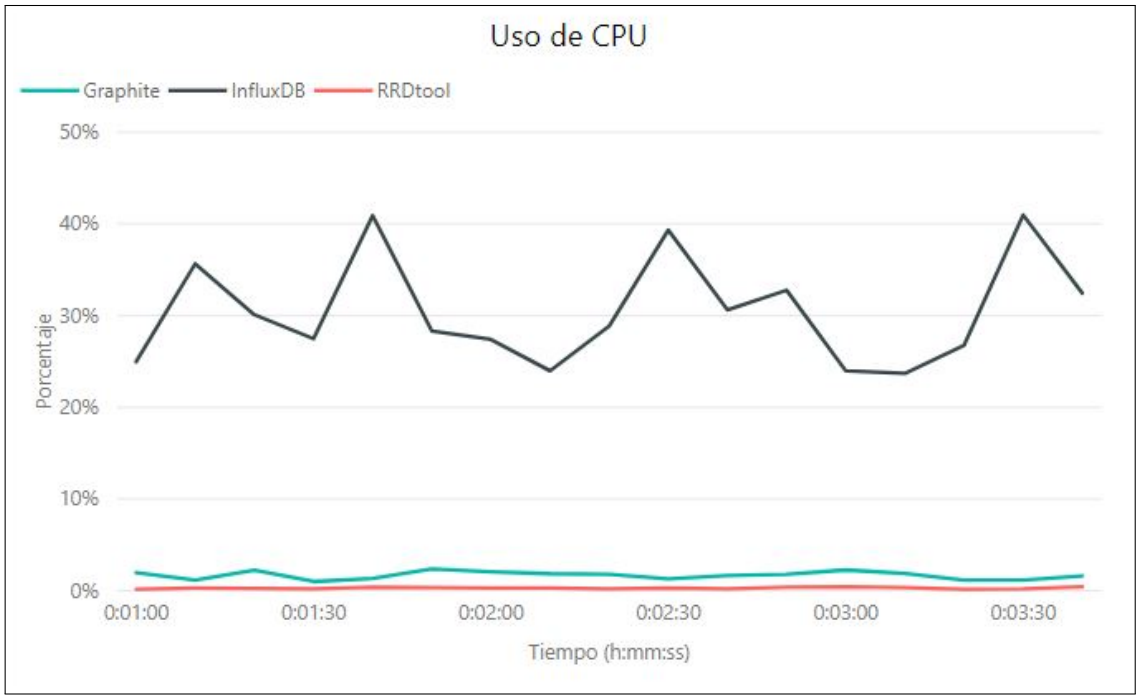


Figura 4.6: Uso de la CPU en función al tiempo de ejecución del segundo escenario presente en el *dataset* 1 (elaboración propia).

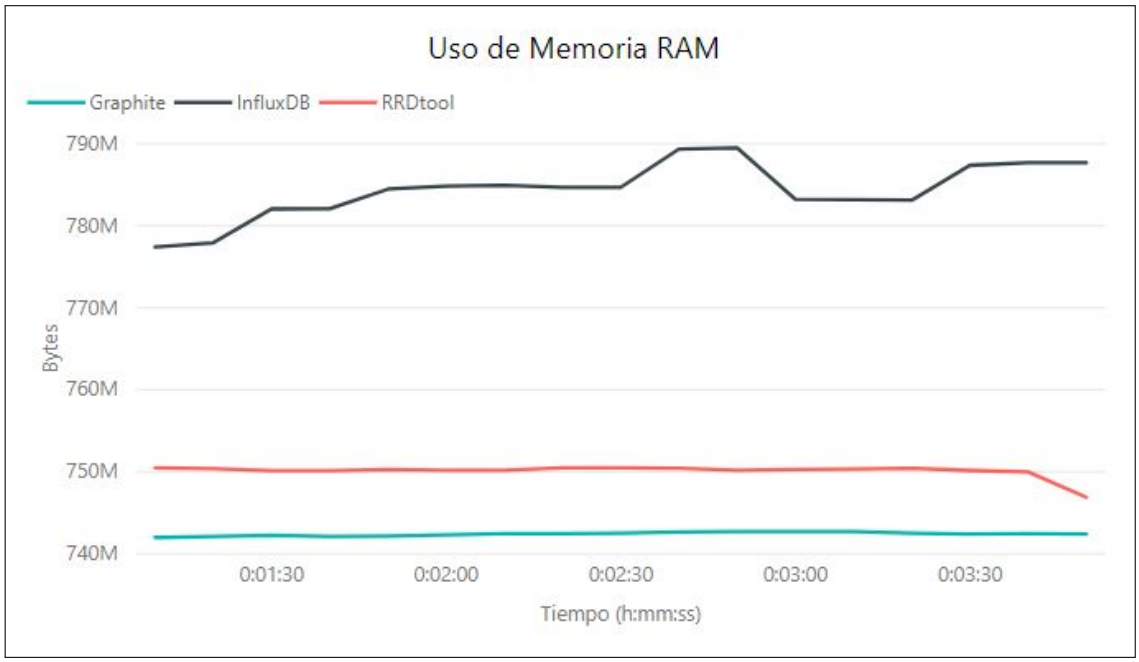
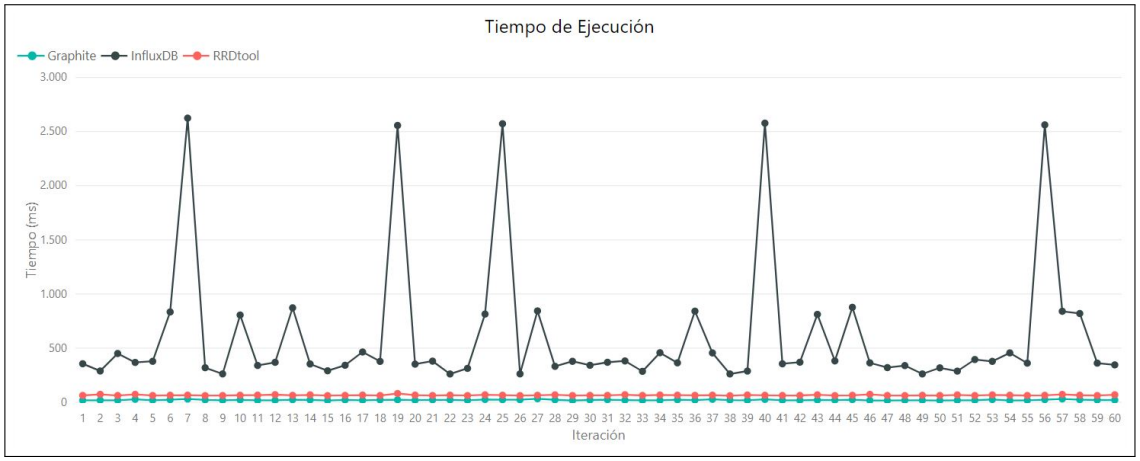
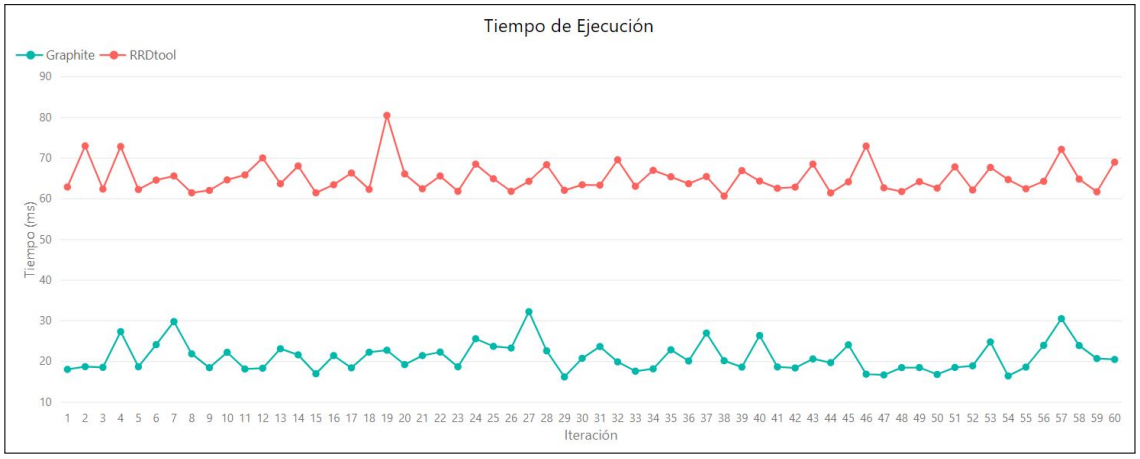


Figura 4.7: Uso de memoria RAM en función al tiempo de ejecución del segundo escenario presente en el *dataset* 1 (elaboración propia).



(a)



(b)

Figura 4.8: (a) Tiempo de ejecución de cada iteración correspondiente al segundo escenario del *dataset* 1 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al segundo escenario del *dataset* 1, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

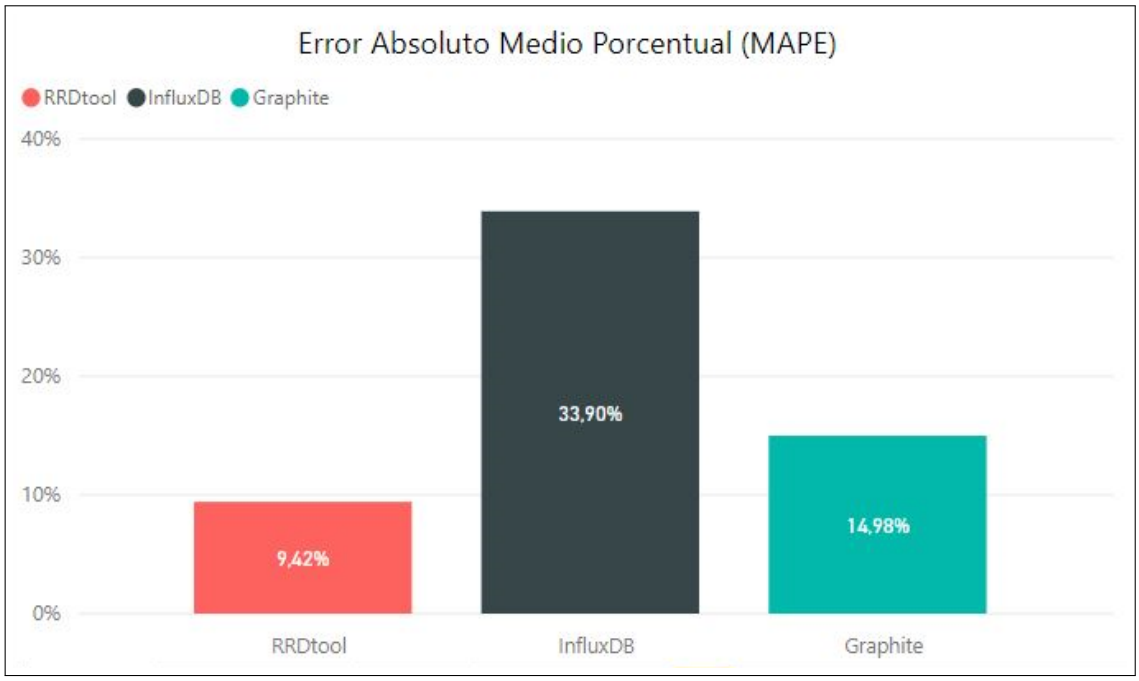


Figura 4.9: Promedio del MAPE correspondiente al segundo escenario del *dataset* 1 (elaboración propia).

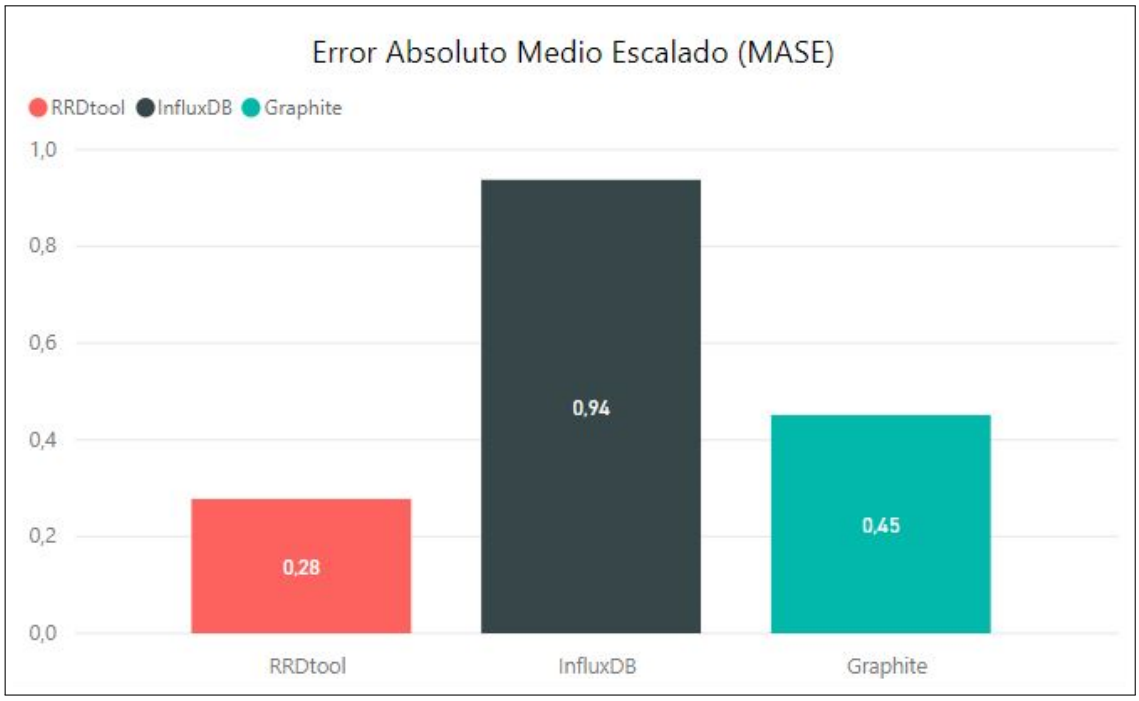


Figura 4.10: Promedio del MASE correspondiente al segundo escenario del *dataset* 1 (elaboración propia).

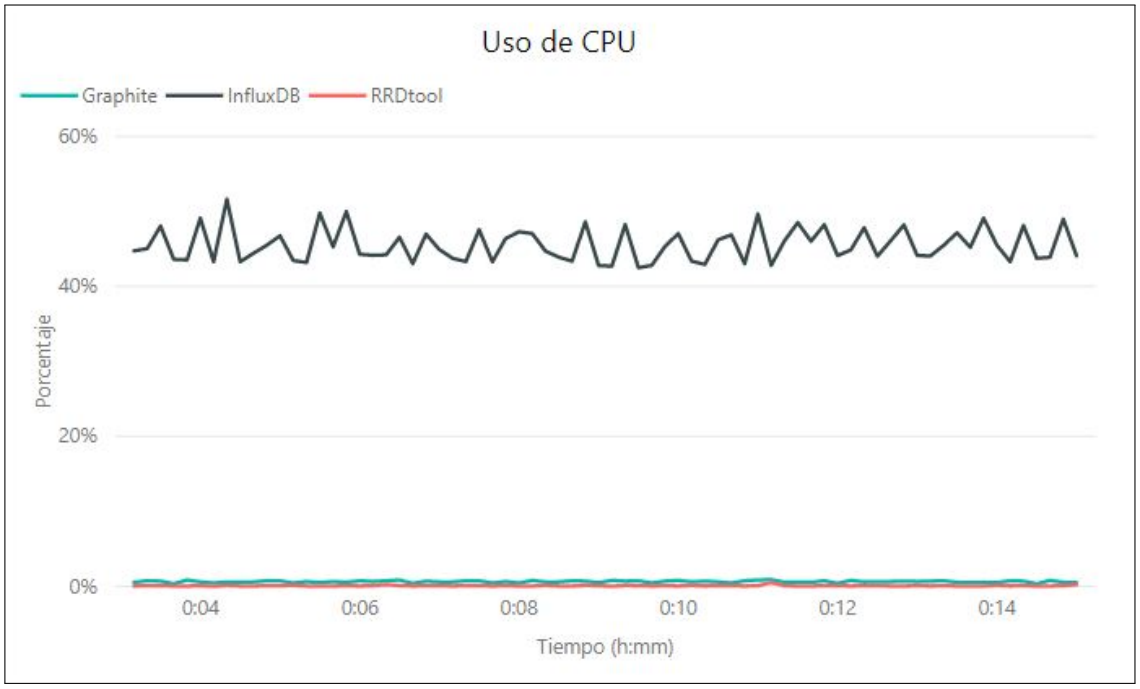
Resumen de resultados en el segundo escenario del *dataset* 1

TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	30,48 %	784,36 MB	616,42 ms	33,90 %	0,94
Graphite	1,69 %	**742,41 MB**	**21,14 ms**	14,98 %	0,45
RRDtool	**0,29 %**	750,09 MB	65,23 ms	**9.42 %**	**0,28**

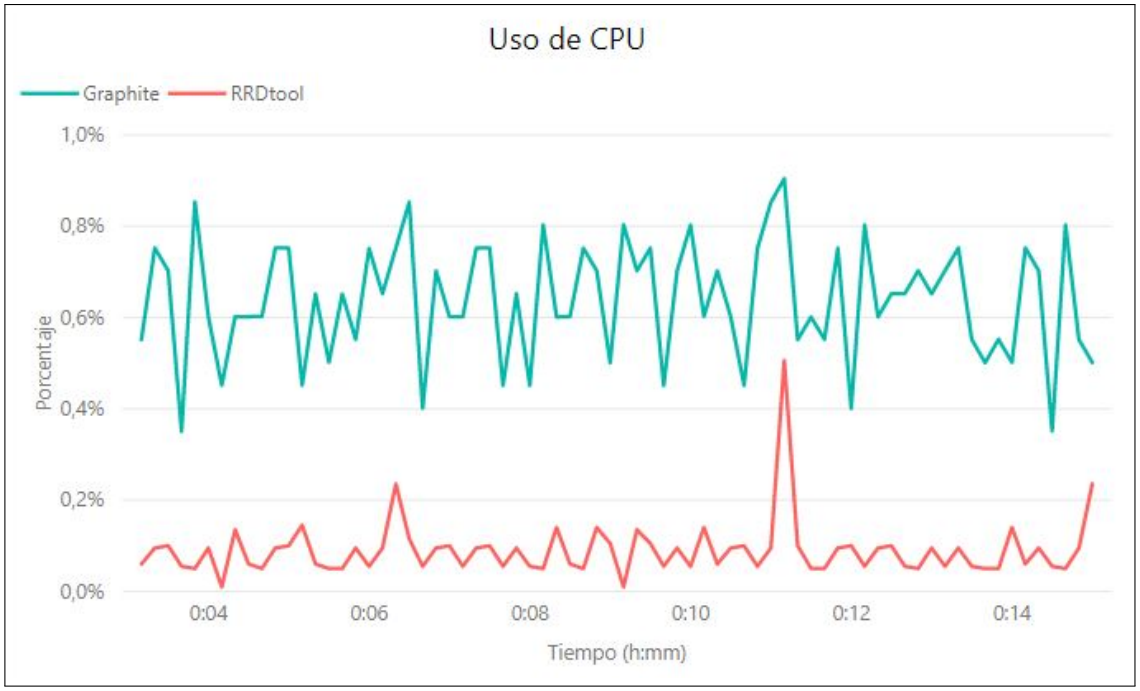
Cuadro 4.2: Resultados promedios de las métricas medidas al ejecutar el segundo escenario del *dataset* 1 (elaboración propia).

De igual manera, se realiza el mismo *test* para el tercer escenario, donde la figura 4.11 muestra que el uso de la CPU no supera el 1 % para el caso de *Graphite* y *RRDtool*; a su vez, *InfluxDB* tiene un uso de la CPU que en promedio es de un 45,51 %. Por otra parte, como se indica en la figura 4.12, el uso de memoria RAM se mantiene constante tanto para *Graphite* como *RRDtool*, en cambio *InfluxDB* presenta un aumento paulatino. De esta manera, el uso promedio de memoria en dichos motores es de 862,07 MB, 827,41 MB y 764,66 MB, respectivamente. El tiempo de ejecución de cada iteración durante el *test* se presenta en la figura 4.13, donde se muestra que el tiempo promedio de ejecución de las consultas predictivas en *InfluxDB* es de 4,55 s, mientras que para *Graphite* y *RRDtool* es de 33,27 ms y 74,18 ms, respectivamente.

En la figura 4.14 se da a conocer que en promedio el MAPE para *InfluxDB*, *Graphite* y *RRDtool* es de 34,37 %, 11,42 % y 0,65 %, respectivamente. Por su parte, la figura 4.15 indica que el promedio del MASE en cada motor es de 1,07, 0,39 y 0,02, respectivamente.



(a)



(b)

Figura 4.11: (a) Uso de la CPU en función al tiempo de ejecución del tercer escenario presente en el *dataset* 1 para los tres motores de bases de datos. (b) Uso de la CPU en función al tiempo de ejecución del tercer escenario presente en el *dataset* 1, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

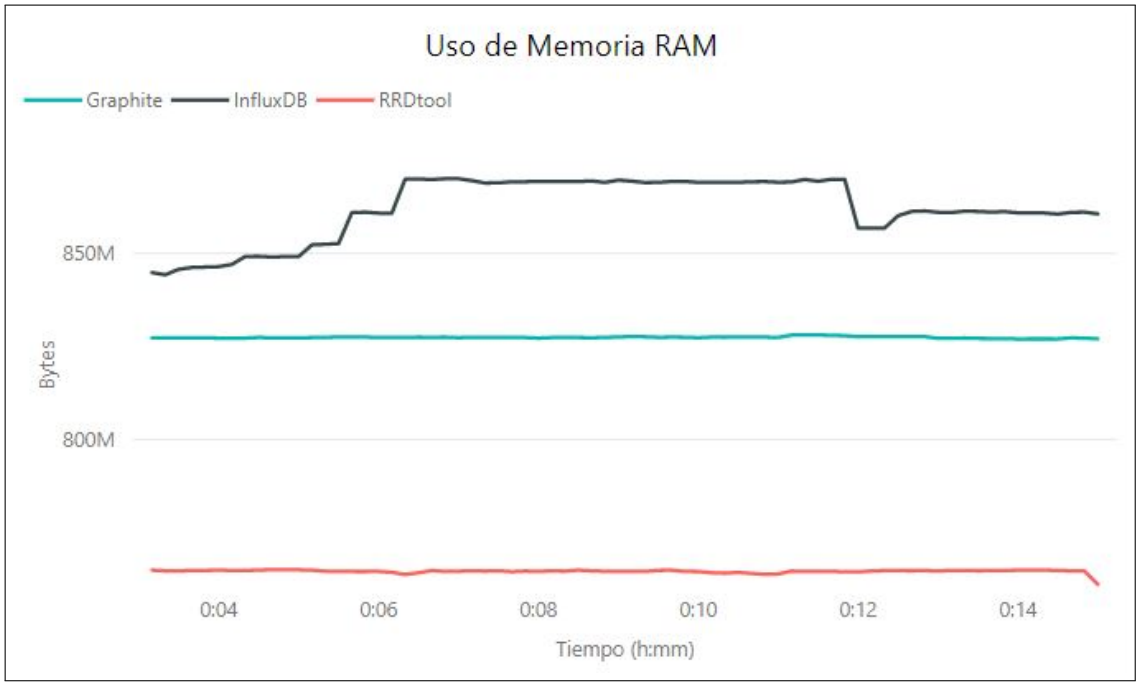
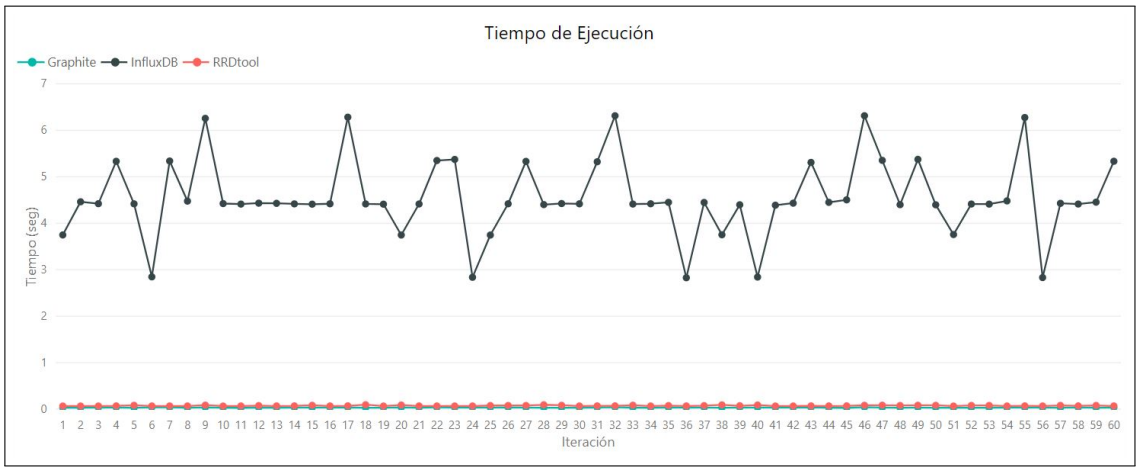
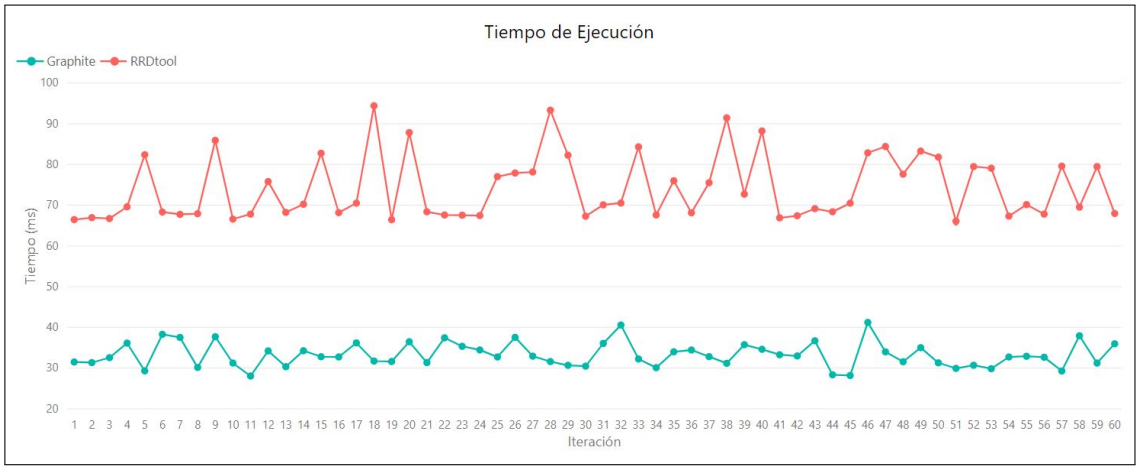


Figura 4.12: Uso de memoria RAM en función al tiempo de ejecución del tercer escenario presente en el *dataset* 1 (elaboración propia).



(a)



(b)

Figura 4.13: (a) Tiempo de ejecución de cada iteración correspondiente al tercer escenario del *dataset* 1 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al tercer escenario del *dataset* 1, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

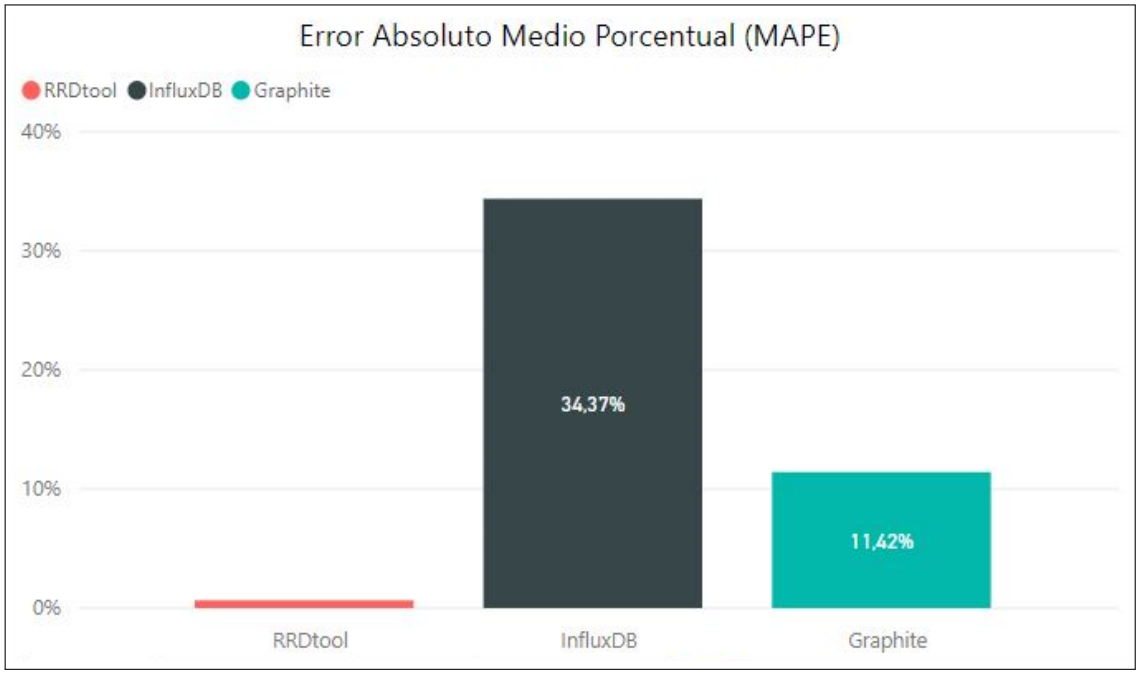


Figura 4.14: Promedio del MAPE correspondiente al tercer escenario del *dataset* 1 (elaboración propia).

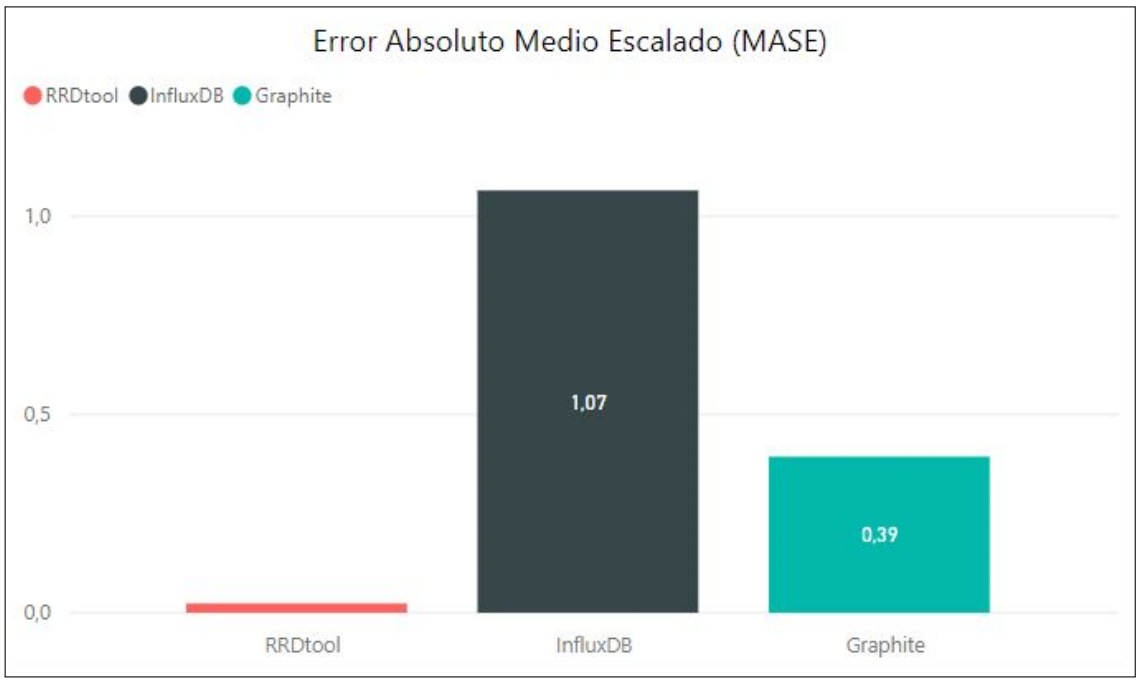


Figura 4.15: Promedio del MASE correspondiente al tercer escenario del *dataset* 1 (elaboración propia).

Resumen de resultados en el tercer escenario del *dataset* 1

TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	45,51 %	862,07 MB	4,55 s	34,37 %	1,07
Graphite	0,64 %	827,41 MB	**33,27 ms**	11,42 %	0,39
RRDtool	**0,10 %**	**764,66 MB**	74,18 ms	**0.65 %**	**0,02**

Cuadro 4.3: Resultados promedios de las métricas medidas al ejecutar el tercer escenario del *dataset* 1 (elaboración propia).

4.3.2. *Dataset 2*

Para el desarrollo de los siguientes tres *tests*, se utiliza el *dataset* 2, del mismo modo al empleado en los *tests* del *set de datos* anterior, y se inicia con la evaluación del primer escenario. De esta manera, las figuras 4.16, 4.17 y 4.18 muestran el rendimiento durante la ejecución del *test*, el cual el comportamiento en el uso de la CPU es similar a lo ya realizado, siendo el uso de la CPU promedio de cada uno de los motores de 37,09 %, 1,65 % y 0,22 % para *InfluxDB*, *Graphite* y *RRDtool*, respectivamente. A su vez, el consumo de memoria RAM se presenta constante en los tres motores de bases de datos. Por otro lado, el tiempo de ejecución en *InfluxDB* oscila frecuentemente entre los valores 270 ms y 3,8 s. Mientras que los otros dos motores presentan un tiempo de ejecución con un corto rango de oscilación, siendo el promedio de sus tiempos de ejecución de 22,74 ms y 84,47 ms correspondiente a *Graphite* y *RRDtool*, respectivamente. Este último presenta un alza en el tiempo de ejecución en algunas de sus iteraciones respecto a las demás.

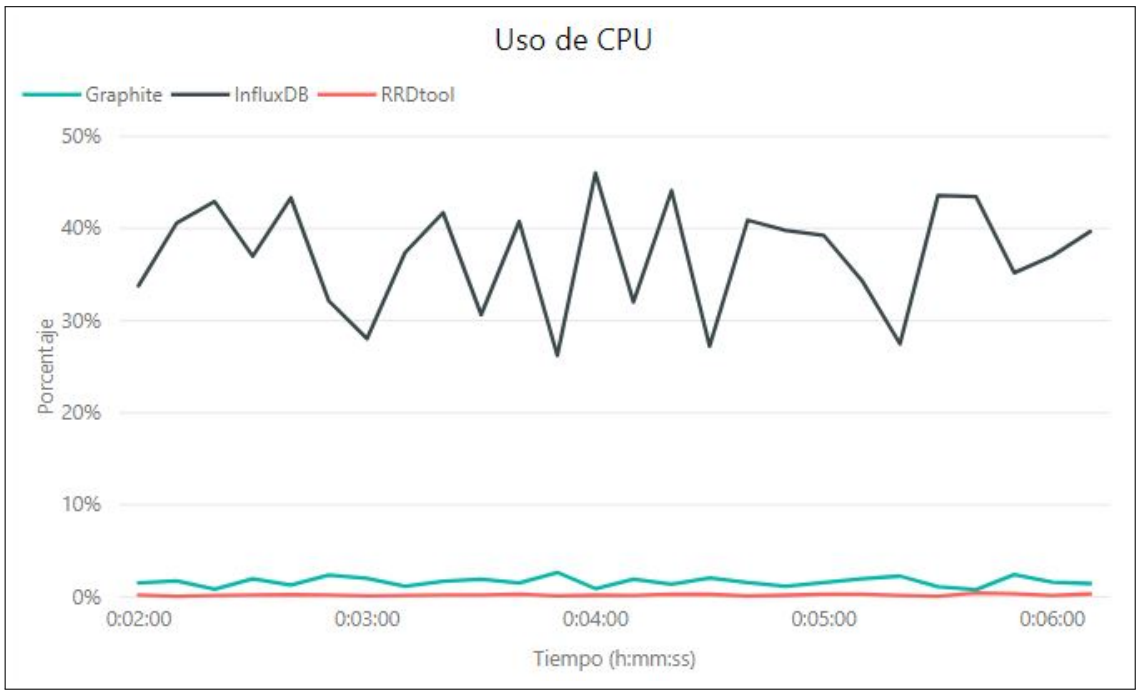


Figura 4.16: Uso de la CPU en función al tiempo de ejecución del primer escenario presente en el *dataset* 2 (elaboración propia).

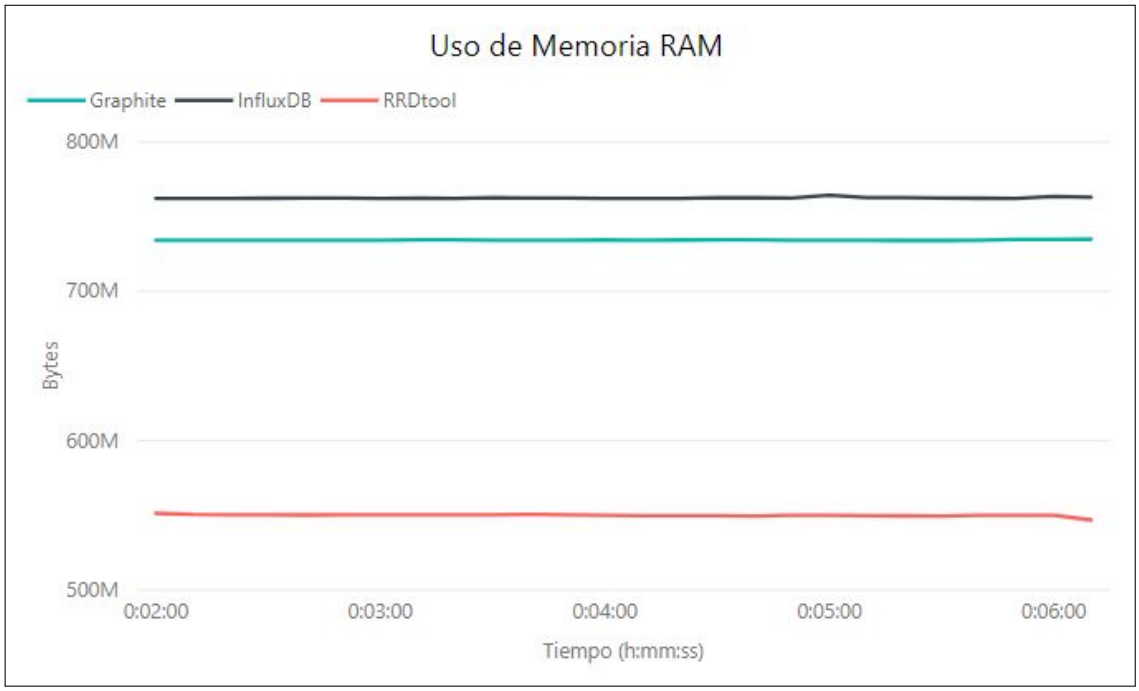
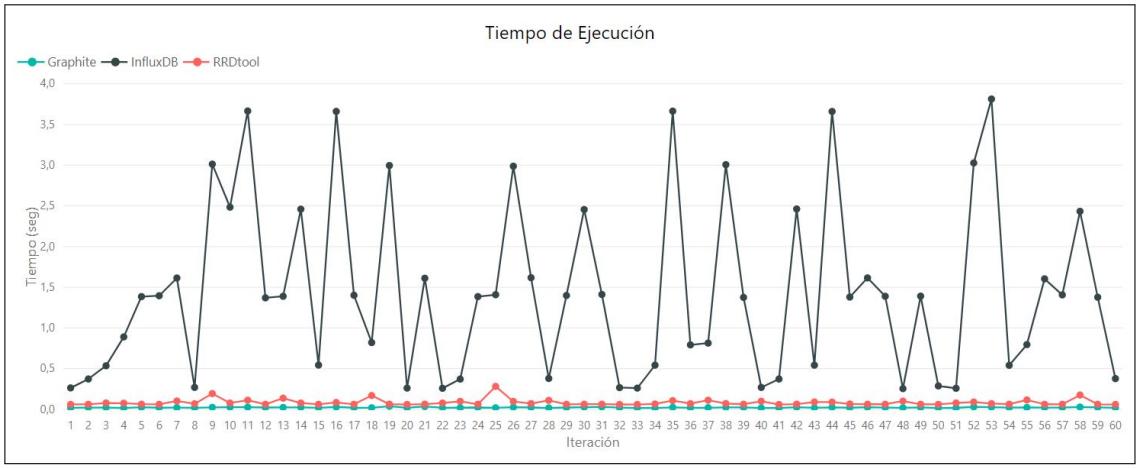
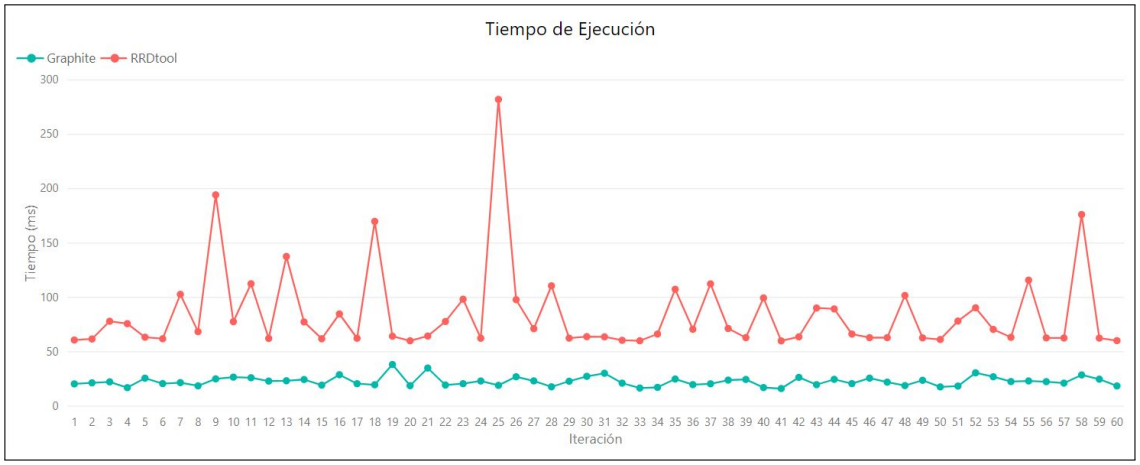


Figura 4.17: Uso de memoria RAM en función al tiempo de ejecución del primer escenario presente en el *dataset* 2 (elaboración propia).



(a)



(b)

Figura 4.18: (a) Tiempo de ejecución de cada iteración correspondiente al primer escenario del *dataset* 2 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al primer escenario del *dataset* 2, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

Las figuras 4.19 y 4.20 dan a conocer la precisión durante el *test*, en las cuales se muestran valores consecuentes a lo presentado en los *tests* anteriores, donde el MAPE promedio más alto está dado por *InfluxDB* con un valor de 29,39 %, del mismo modo que el MASE con un valor de 0,89. Sin embargo, *RRDtool* cuenta con los valores más bajos en ambas mediciones de error, siendo estos promedios de 4,79 % y 0,15, respectivamente.

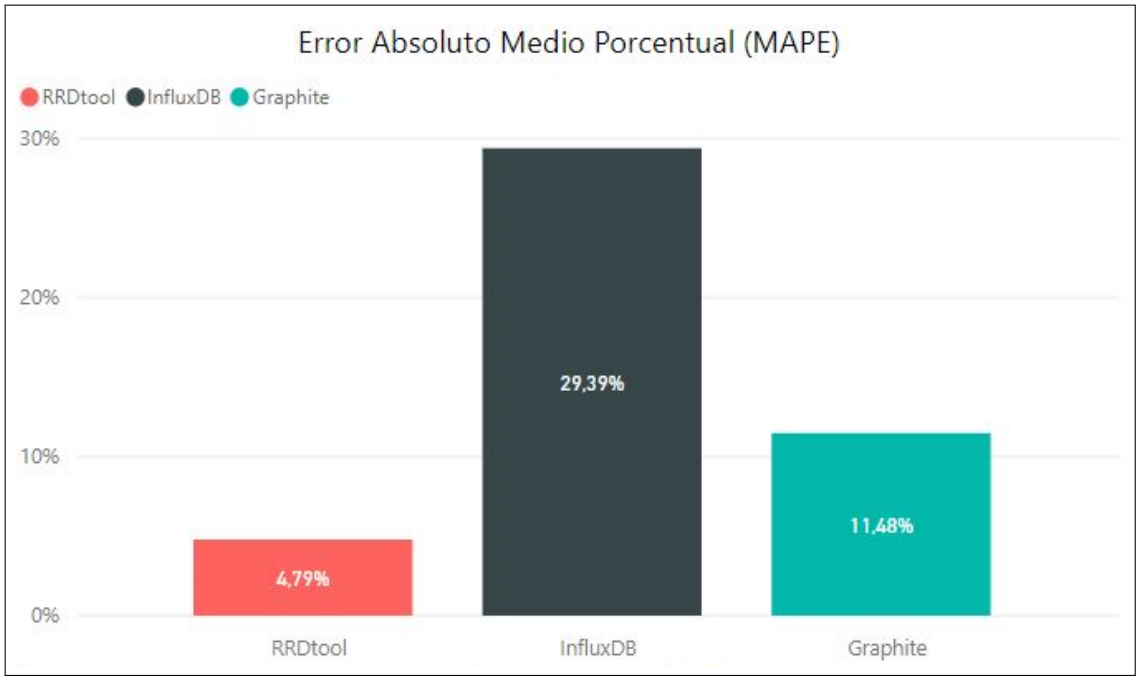


Figura 4.19: Promedio del MAPE correspondiente al primer escenario del *dataset* 2 (elaboración propia).

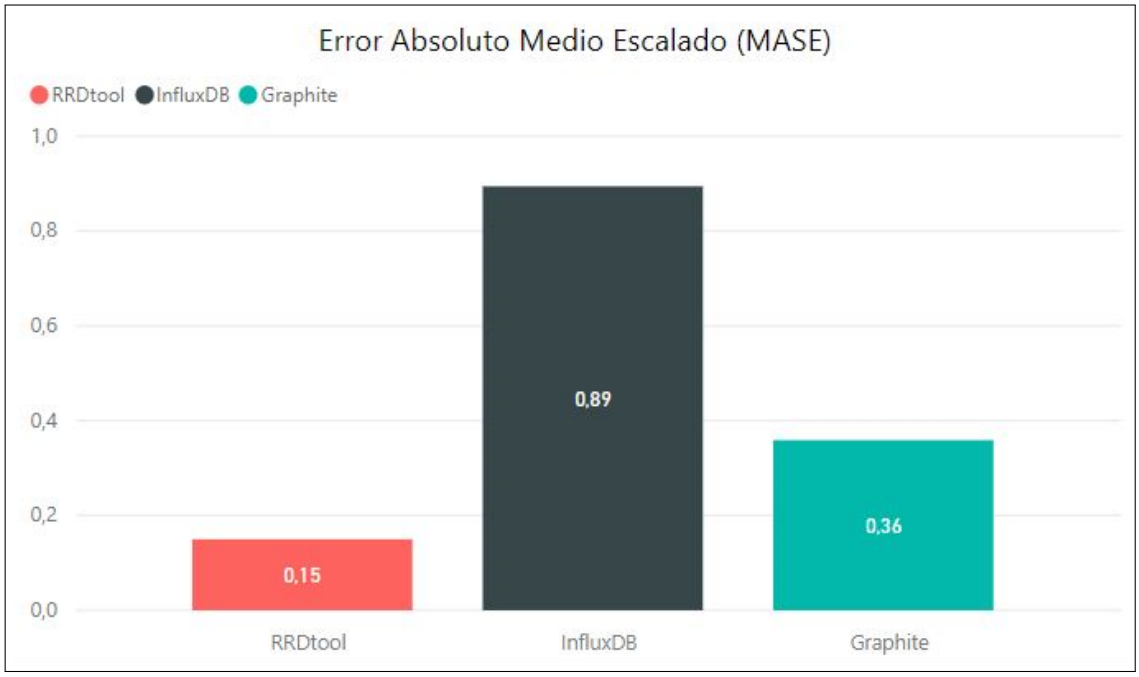


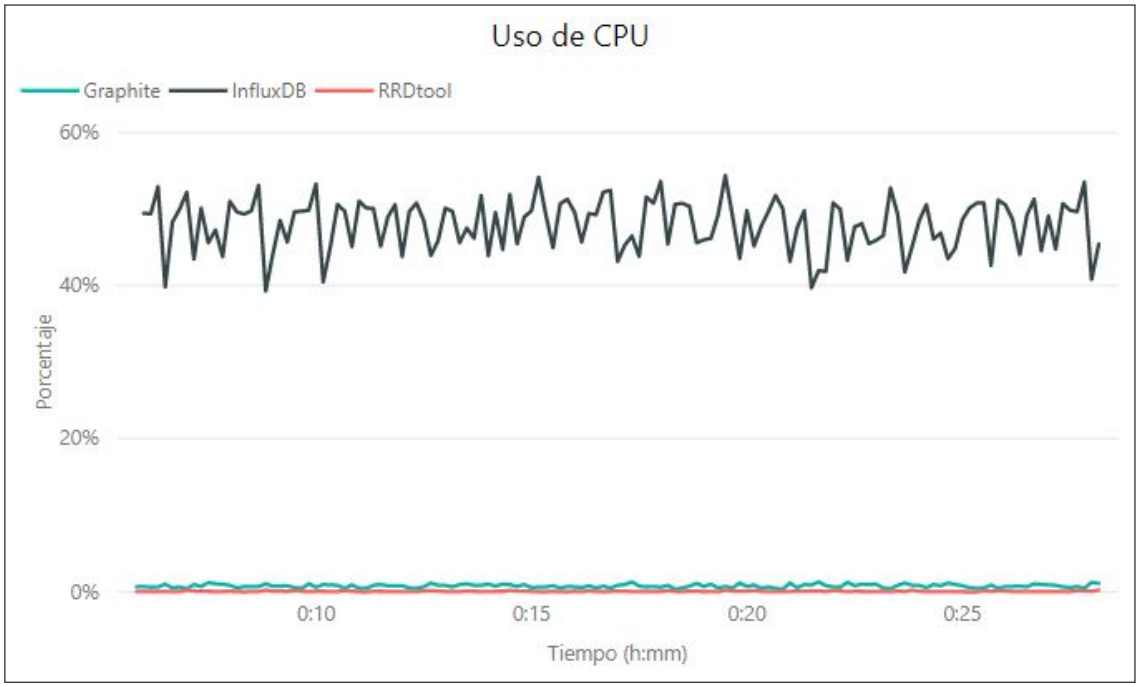
Figura 4.20: Promedio del MASE correspondiente al primer escenario del *dataset* 2 (elaboración propia).

Resumen de resultados en el primer escenario del *dataset* 2

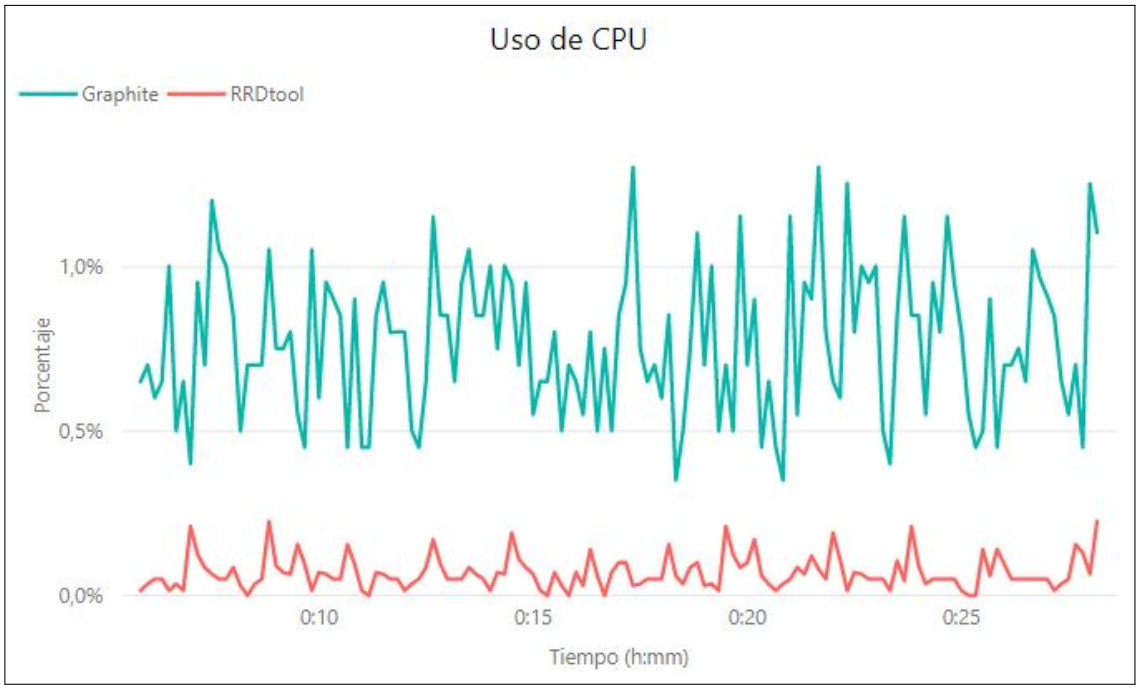
TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	37,09 %	762,42 MB	1,43 s	29,39 %	0,89
Graphite	1,65 %	734,22 MB	**22,74 ms**	11,48 %	0,36
RRDtool	**0,22 %**	**550,09 MB**	84,47 ms	**4,79 %**	**0,15**

Cuadro 4.4: Resultados promedios de las métricas medidas al ejecutar el primer escenario del *dataset* 2 (elaboración propia).

Posteriormente, se realiza el *test* para el segundo escenario del *dataset* 2, cuyos resultados dan cuenta de un comportamiento similar a los resultados correspondientes al primer escenario. Las figuras 4.21, 4.22 y 4.23 muestran el uso de la CPU, uso de la memoria RAM y tiempo de ejecución, respectivamente, durante el *test*; mientras que las figuras 4.24 y 4.25 presentan la precisión mediante las mediciones del error MAPE y MASE.



(a)



(b)

Figura 4.21: (a) Uso de la CPU en función al tiempo de ejecución del segundo escenario presente en el *dataset* 2 para los tres motores de bases de datos. (b) Uso de la CPU en función al tiempo de ejecución del segundo escenario presente en el *dataset* 2, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

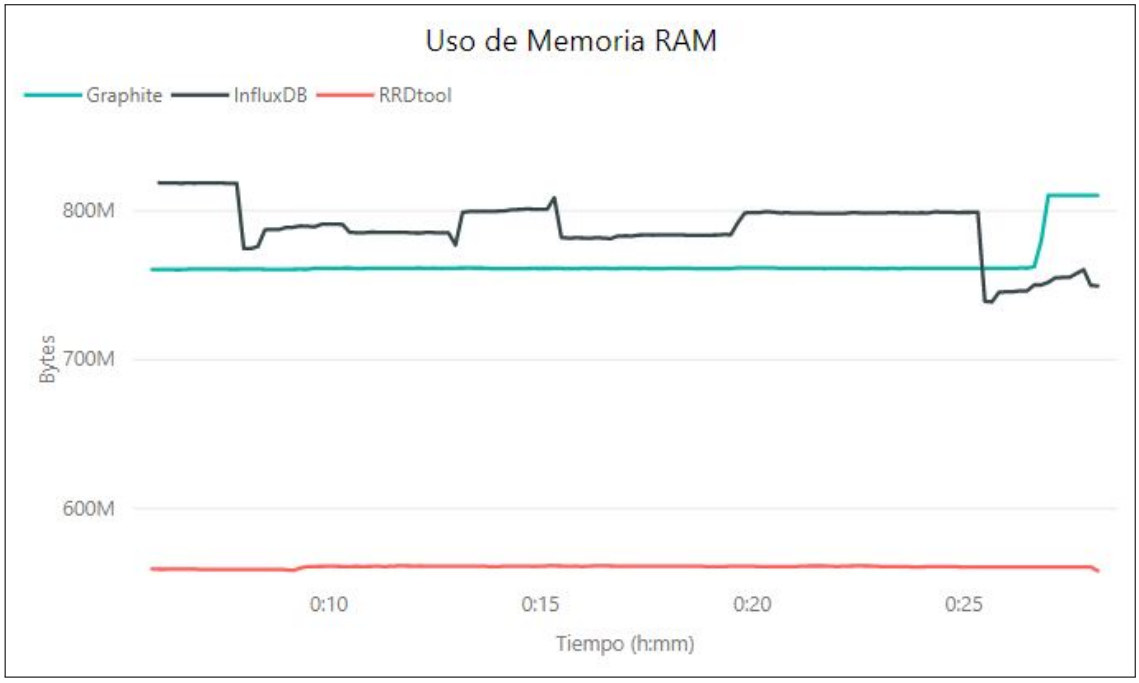
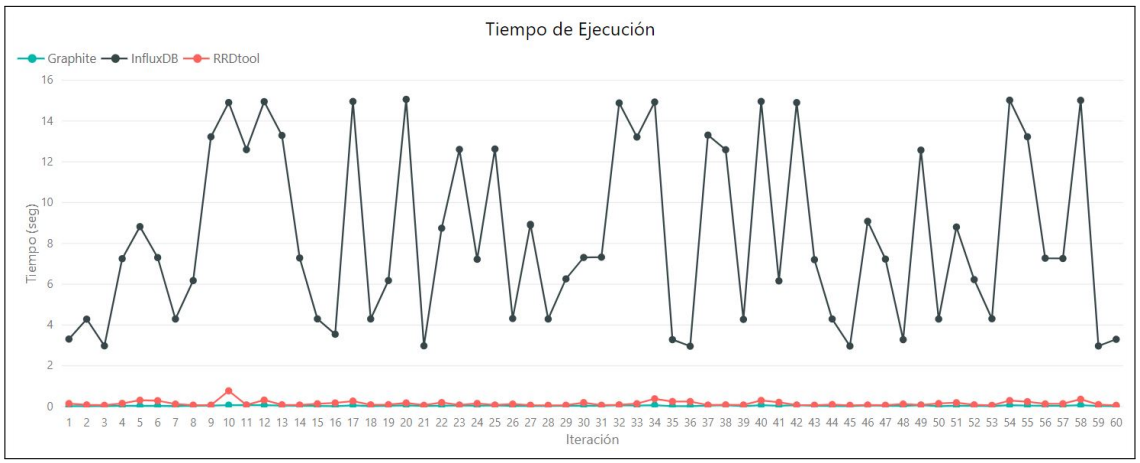
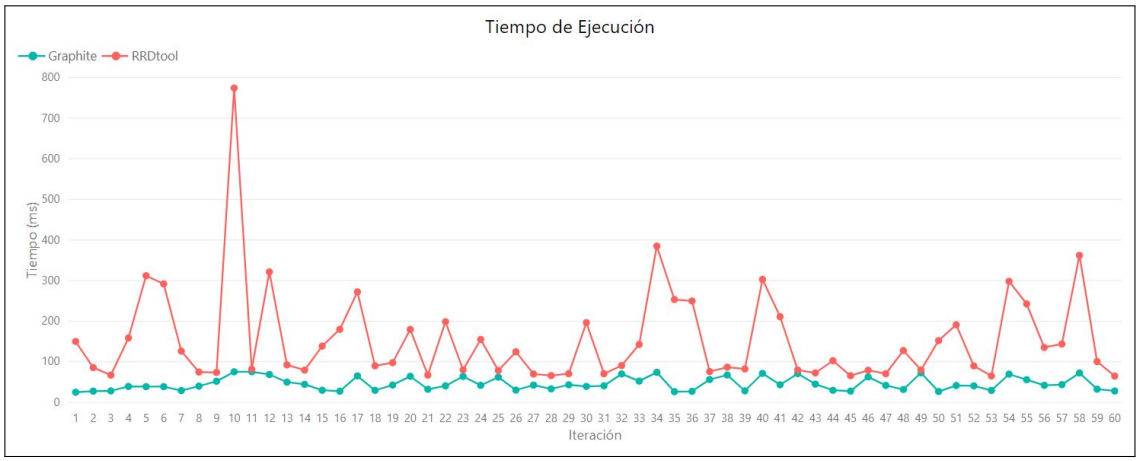


Figura 4.22: Uso de memoria RAM en función al tiempo de ejecución del segundo escenario presente en el *dataset* 2 (elaboración propia).



(a)



(b)

Figura 4.23: (a) Tiempo de ejecución de cada iteración correspondiente al segundo escenario del *dataset* 2 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al segundo escenario del *dataset* 2, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

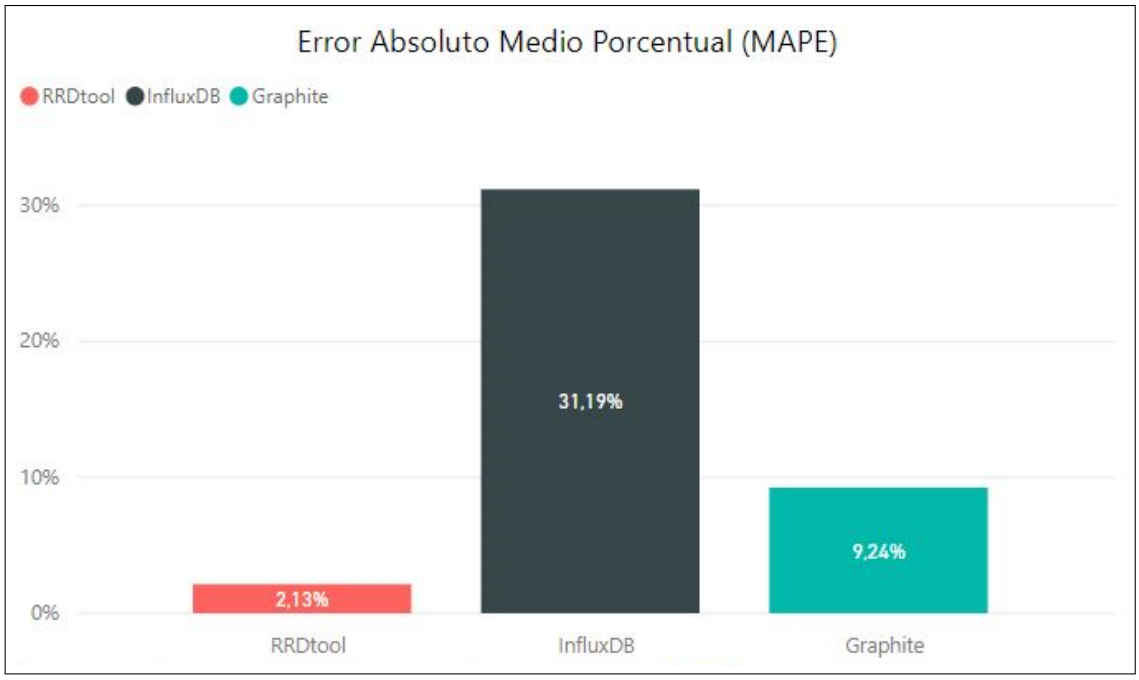


Figura 4.24: Promedio del MAPE correspondiente al segundo escenario del *dataset* 2 (elaboración propia).

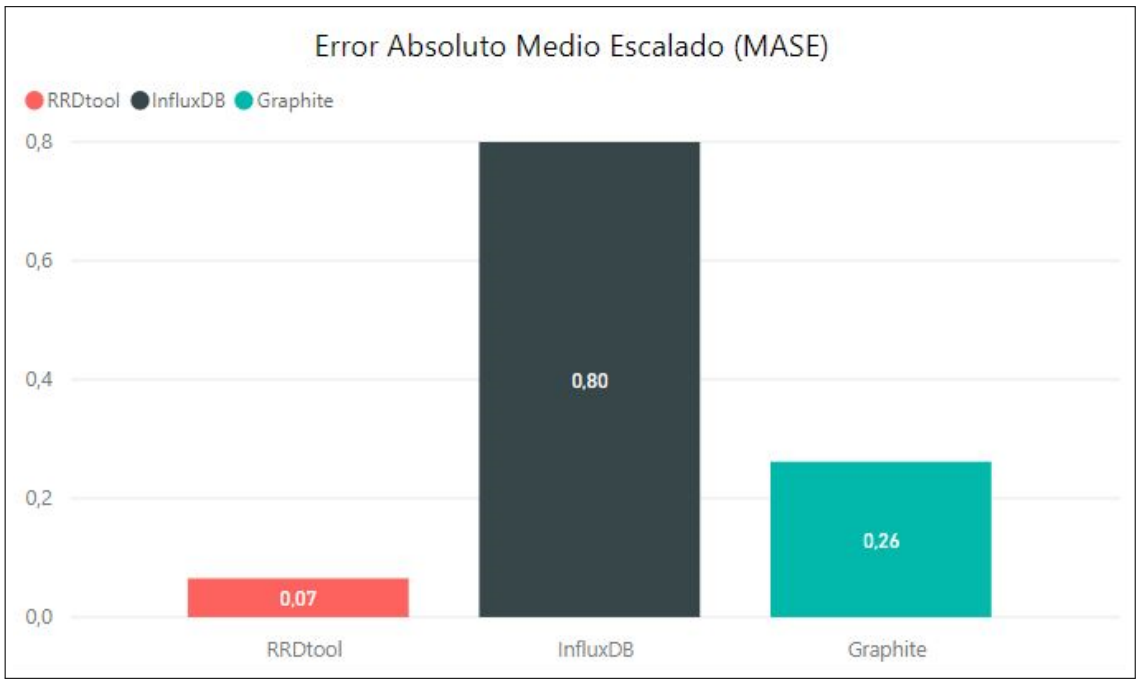


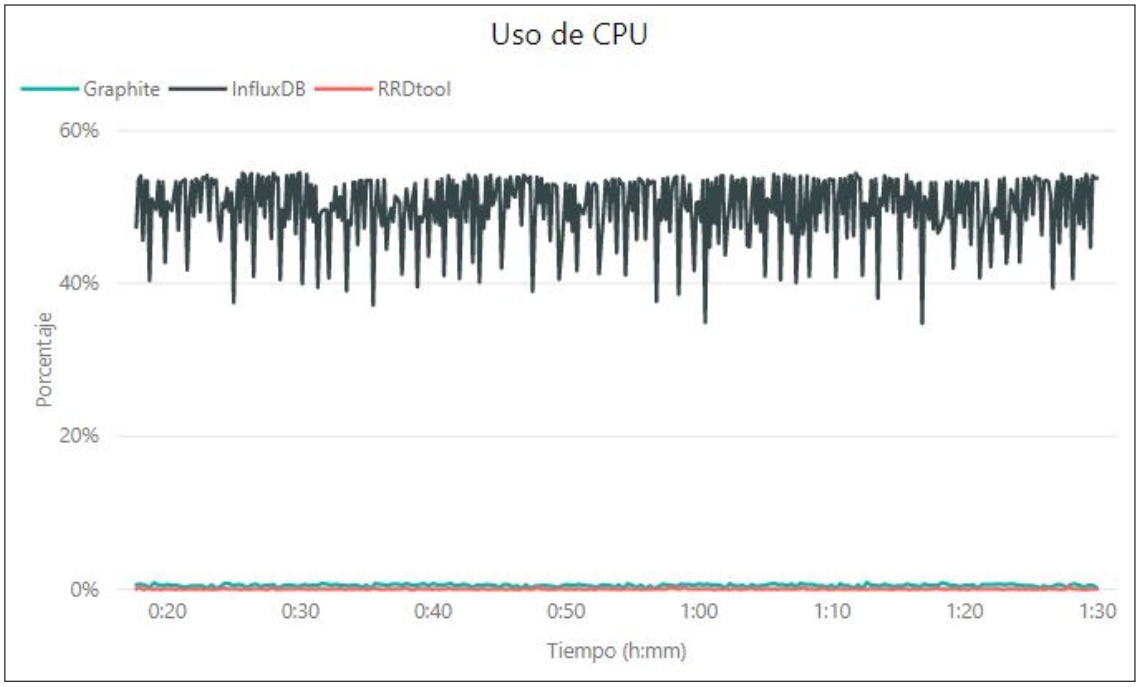
Figura 4.25: Promedio del MASE correspondiente al segundo escenario del *dataset* 2 (elaboración propia).

Resumen de resultados en el segundo escenario del *dataset* 2

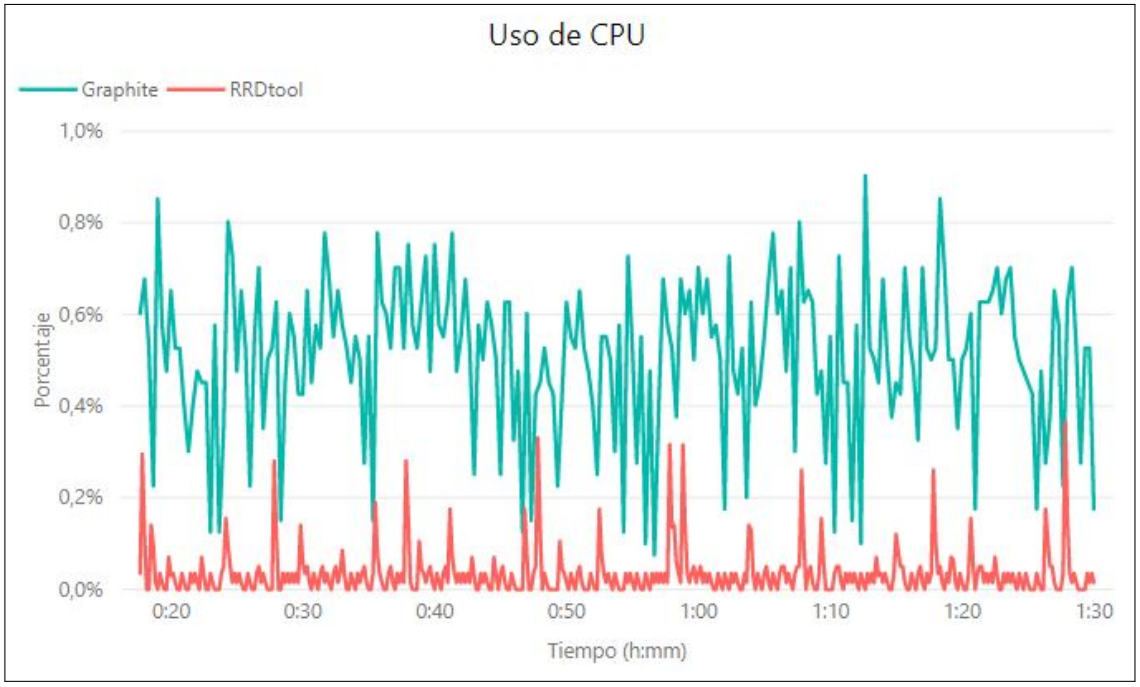
TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	47,94 %	788,48 MB	8,36 s	31,19 %	0,80
Graphite	0,77 %	764,32 MB	**45,64 ms**	9,24 %	0,26
RRDtool	**0,07 %**	**560,88 MB**	153,74 ms	**2,13 %**	**0,07**

Cuadro 4.5: Resultados promedios de las métricas medidas al ejecutar el segundo escenario del *dataset* 2 (elaboración propia).

Finalmente, se ejecuta el tercer escenario del *dataset* 2, siendo sus resultados consecuentes con lo que se ha presentado anteriormente. Las figuras 4.26, 4.27 y 4.28 muestran el uso de la CPU, uso de la memoria RAM y tiempo de ejecución, respectivamente, durante la ejecución del *test*; donde se visualiza una alza en el uso de la CPU tras cada escenario por parte de *InfluxDB*. Además de presentar un aumento en los valores correspondientes a las mediciones de error MAPE y MASE de dicho motor de base de datos, cuyos valores son de 63,53 % y 1,48, respectivamente, como se presenta en las figuras 4.29 y 4.30.



(a)



(b)

Figura 4.26: (a) Uso de la CPU en función al tiempo de ejecución del tercer escenario presente en el *dataset* 2 para los tres motores de bases de datos. (b) Uso de la CPU en función al tiempo de ejecución del tercer escenario presente en el *dataset* 2, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

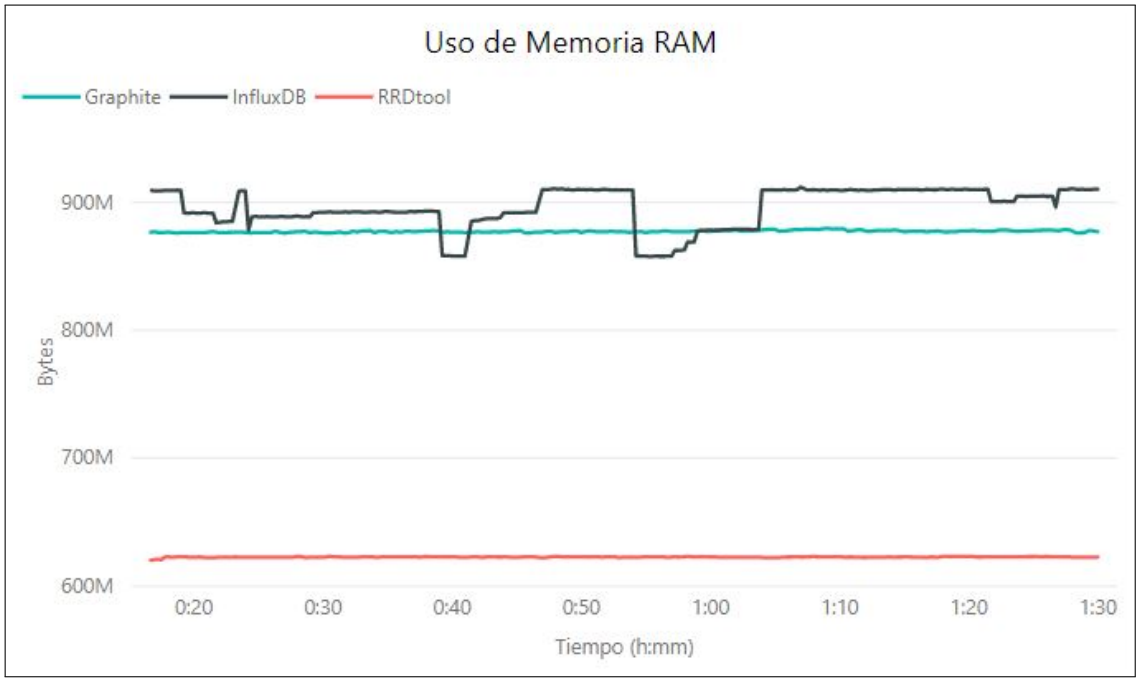
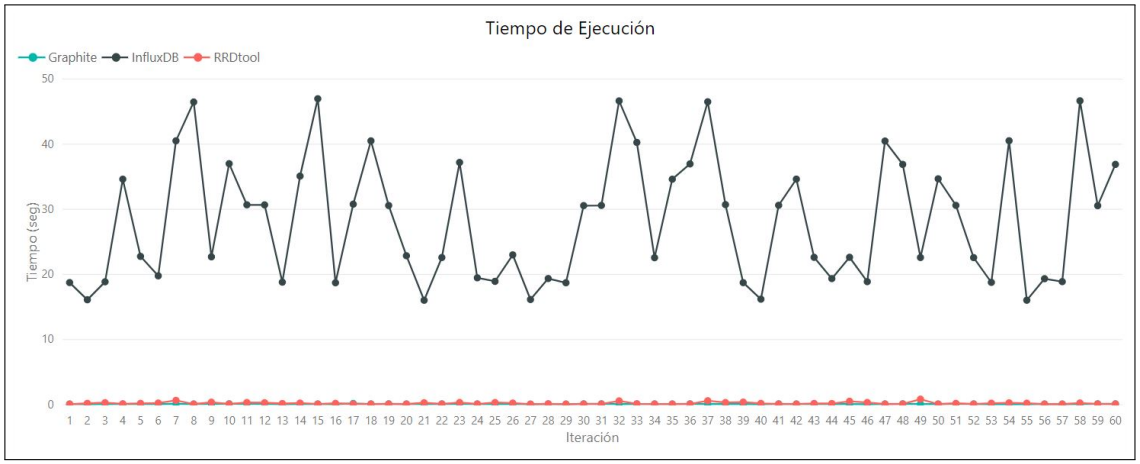
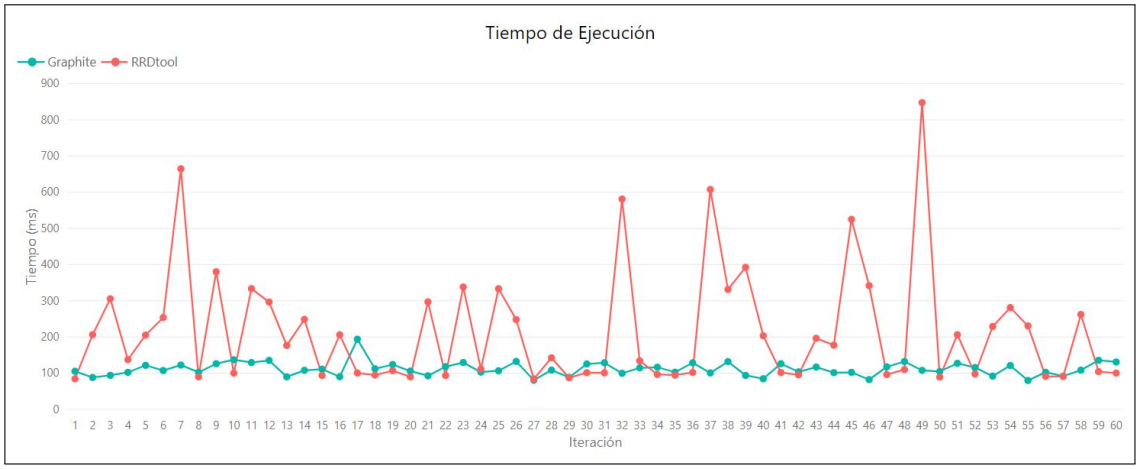


Figura 4.27: Uso de memoria RAM en función al tiempo de ejecución del tercer escenario presente en el *dataset* 2 (elaboración propia).



(a)



(b)

Figura 4.28: (a) Tiempo de ejecución de cada iteración correspondiente al tercer escenario del *dataset* 2 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al tercer escenario del *dataset* 2, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

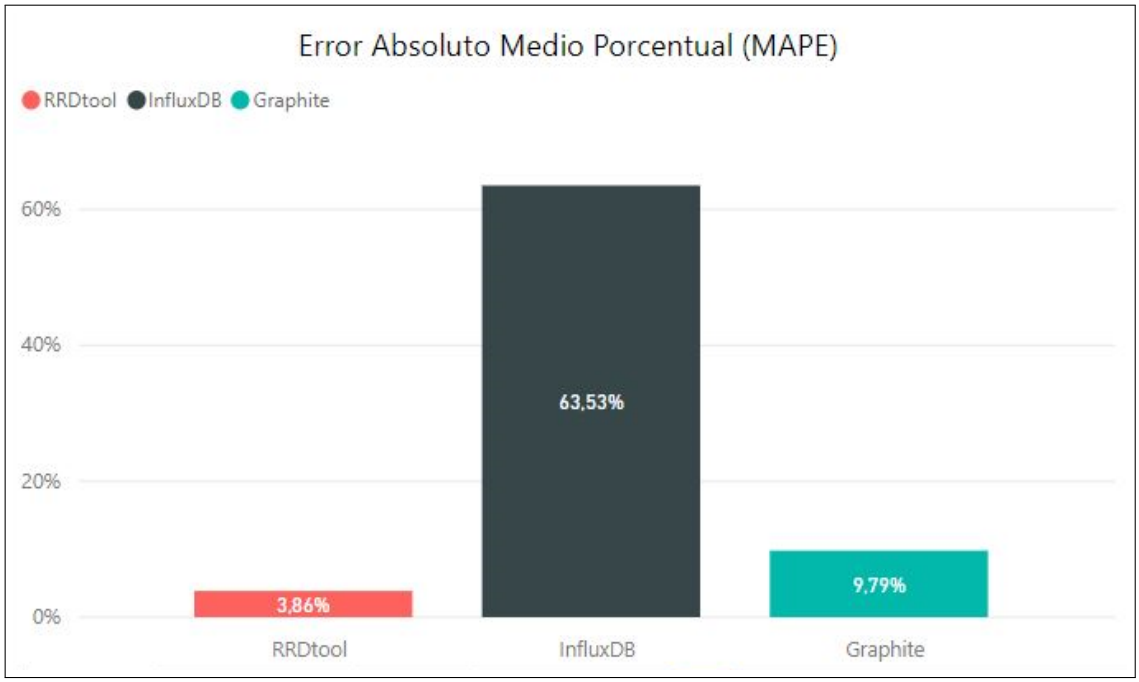


Figura 4.29: Promedio del MAPE correspondiente al tercer escenario del *dataset* 2 (elaboración propia).

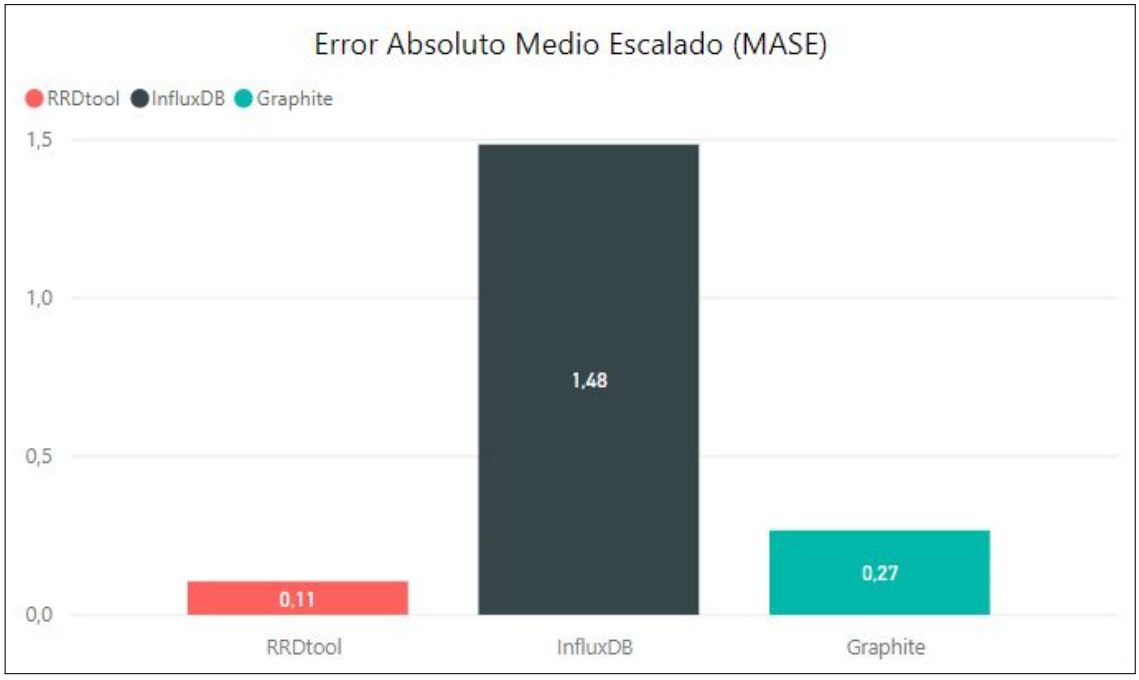


Figura 4.30: Promedio del MASE correspondiente al tercer escenario del *dataset* 2 (elaboración propia).

Resumen de resultados en el tercer escenario del *dataset* 2

TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	50,00 %	895,95 MB	28,23 s	63,53 %	1,48
Graphite	0,51 %	877,27 MB	**111,69 ms**	9,79 %	0,27
RRDtool	**0,03 %**	**622,80 MB**	215,61 ms	**3,86 %**	**0,11**

Cuadro 4.6: Resultados promedios de las métricas medidas al ejecutar el tercer escenario del *dataset* 2 (elaboración propia).

4.3.3. *Dataset* 3

En los últimos tres *tests* a realizar se utiliza el *dataset* 3. De igual modo a las evaluaciones anteriores, se inicia con el desarrollo del primer escenario. La figura 4.31 muestra el uso de la CPU promedio de los tres motores de bases de datos, donde *InfluxDB* presenta un valor de 30,45 %, mientras que *Graphite* y *RRDtool* no superan los 2 %. La figura 4.32 grafica que el consumo de la memoria RAM se mantiene constante, cuyos valores promedios son de 979,24 MB, 729,92 MB y 814,57 MB para *InfluxDB*, *Graphite* y *RRDtool*, respectivamente. El tiempo de ejecución, como se indica en la figura 4.33, presenta oscilaciones en *InfluxDB*, mientras que para los otros dos motores se mantiene constante con algunas variaciones entre cada iteración.

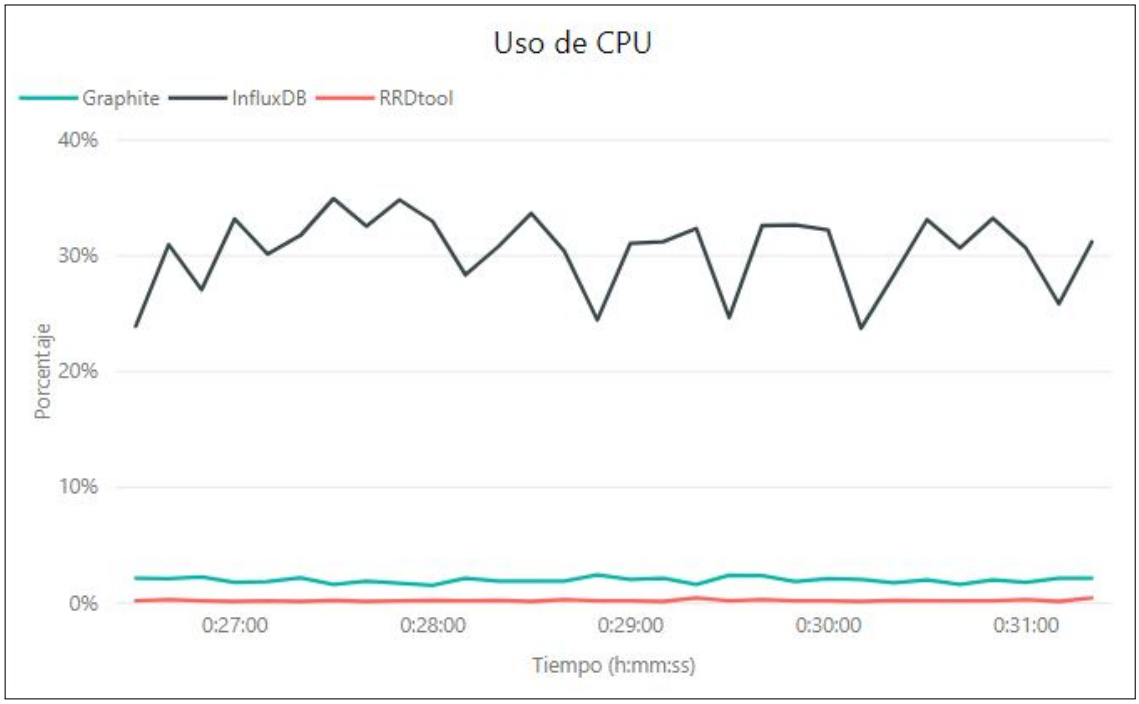


Figura 4.31: Uso de la CPU en función al tiempo de ejecución del primer escenario presente en el *dataset* 3 (elaboración propia).

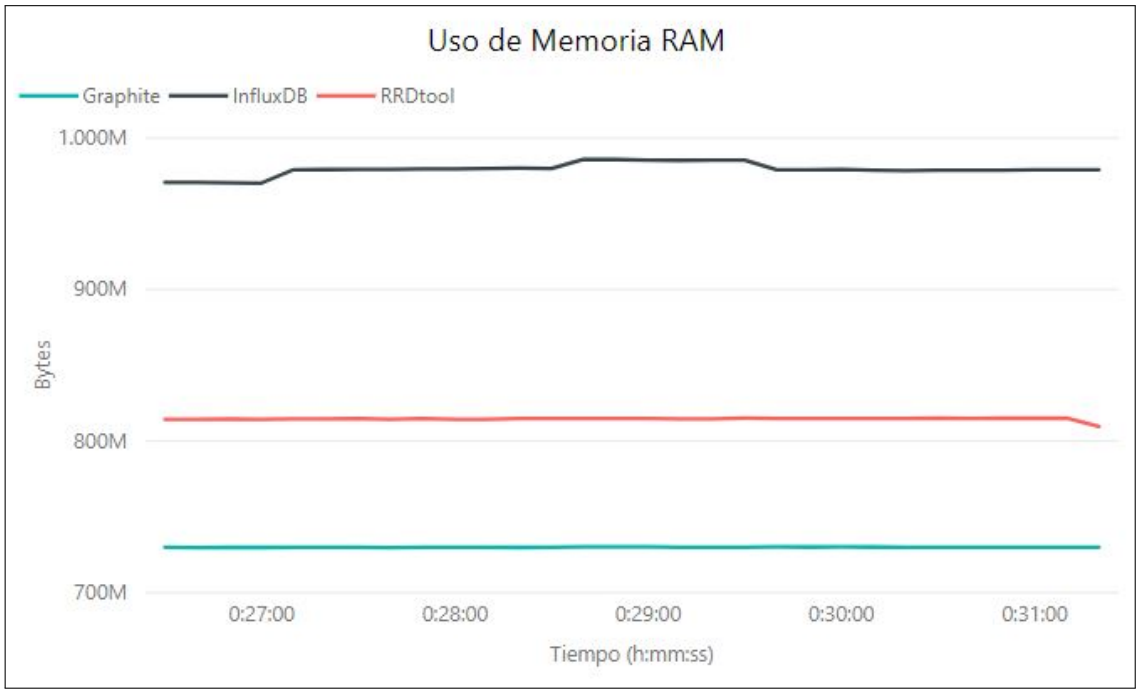
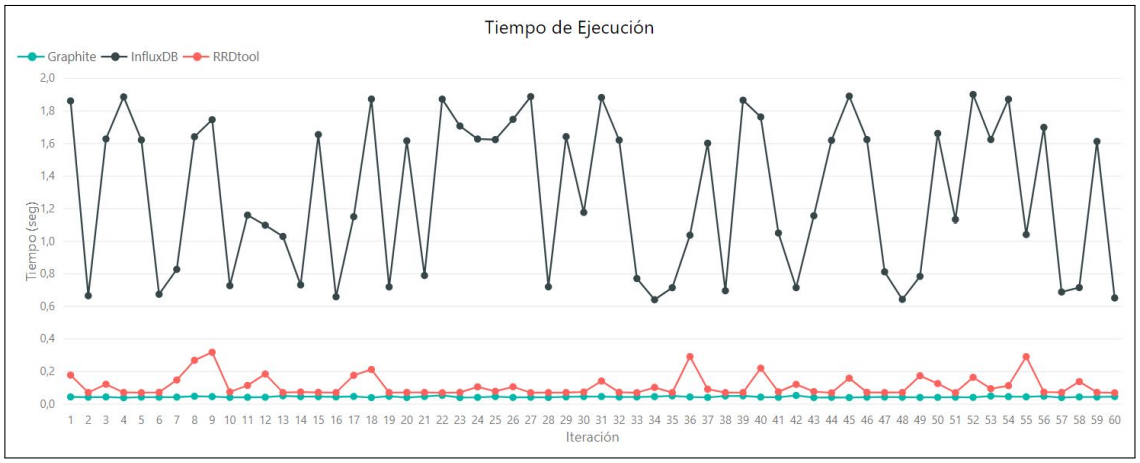
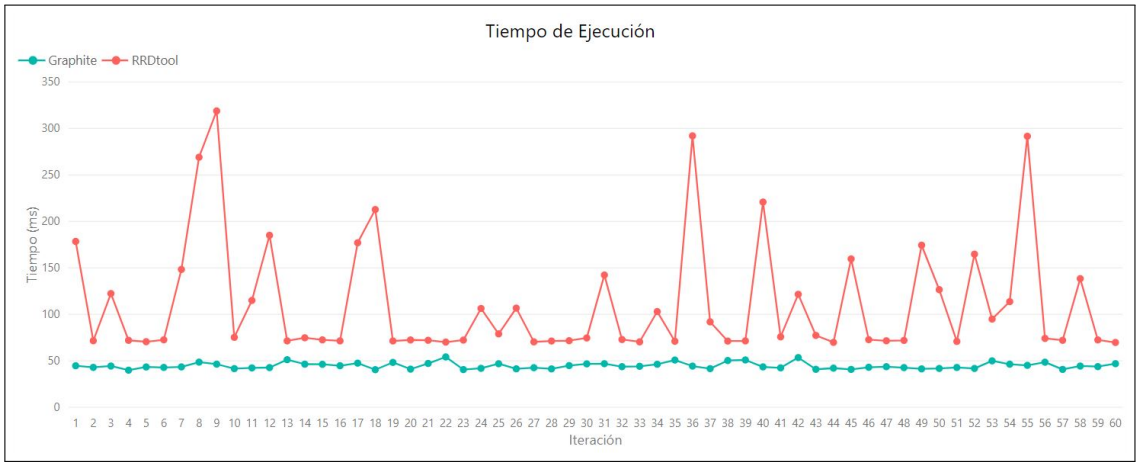


Figura 4.32: Uso de memoria RAM en función al tiempo de ejecución del primer escenario presente en el *dataset* 3 (elaboración propia).



(a)



(b)

Figura 4.33: (a) Tiempo de ejecución de cada iteración correspondiente al primer escenario del *dataset* 3 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al primer escenario del *dataset* 3, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

Por otra parte, las mediciones de error se muestran en las figuras 4.34 y 4.35, donde *InfluxDB* presenta los valores promedios más elevados, de MAPE y MASE, correspondientes a 130,14 % y 1,44, respectivamente. Los menores valores promedios se registran para *Graphite*.

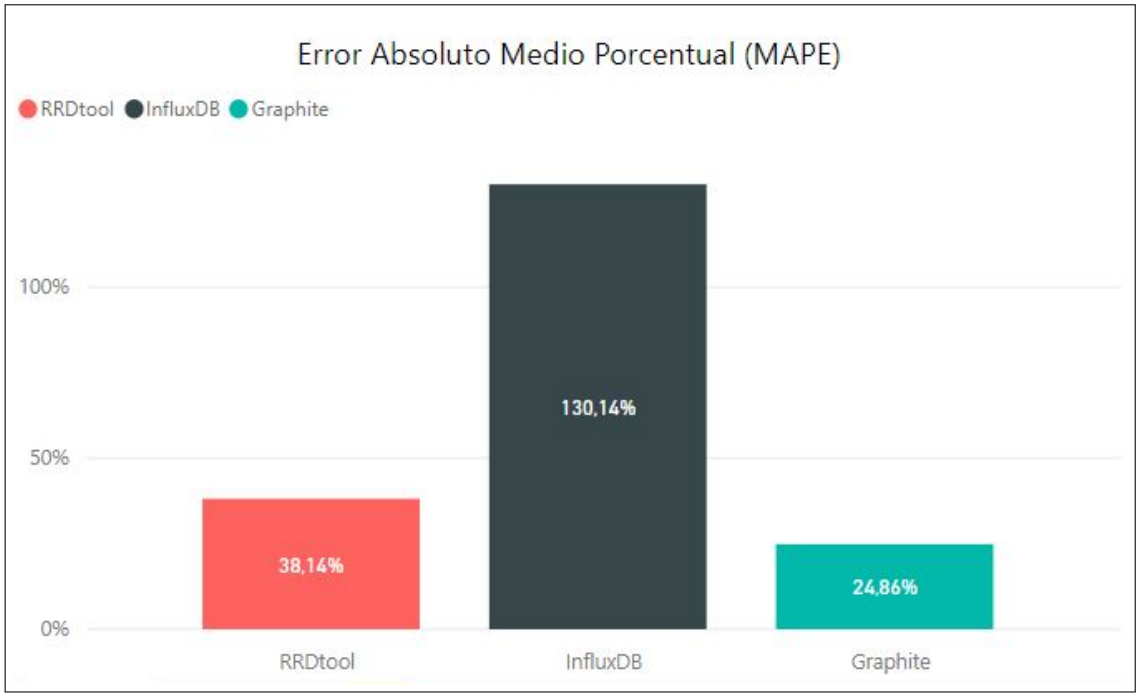


Figura 4.34: Promedio del MAPE correspondiente al primer escenario del *dataset* 3 (elaboración propia).

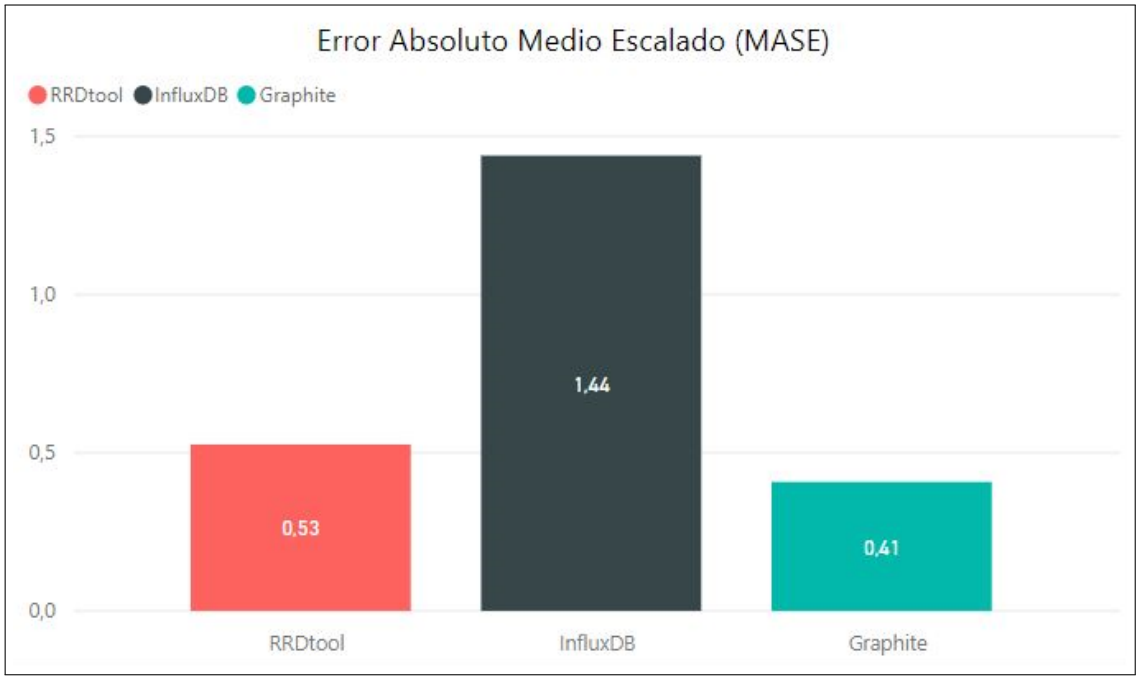


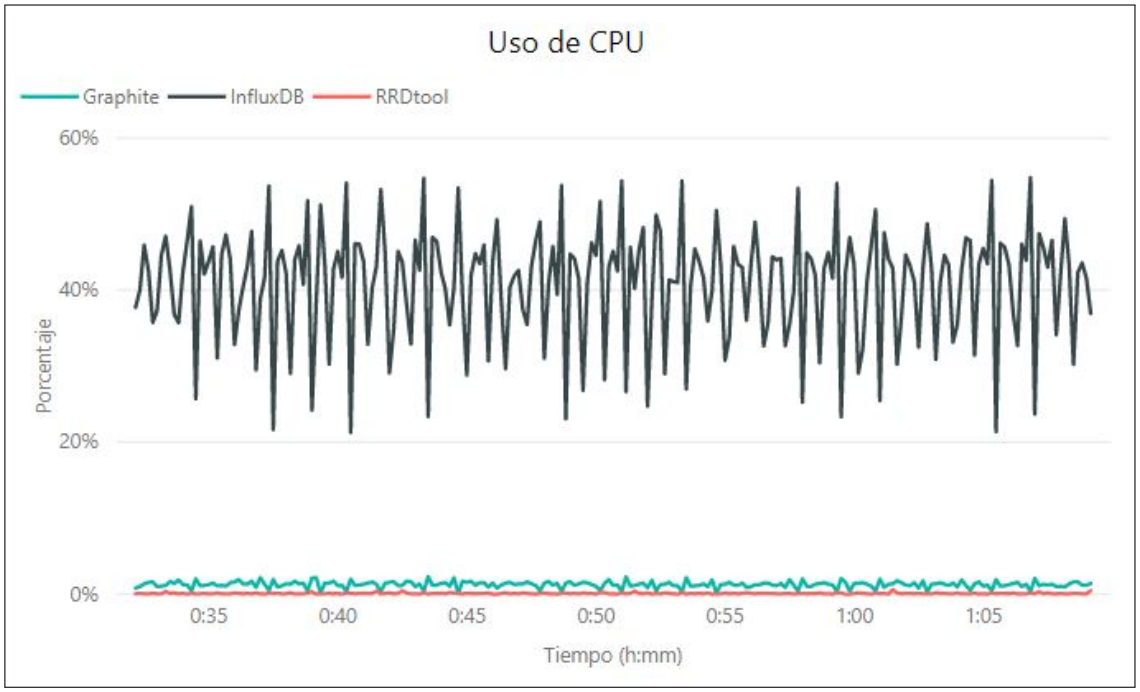
Figura 4.35: Promedio del MASE correspondiente al primer escenario del *dataset* 3 (elaboración propia).

Resumen de resultados en el primer escenario del *dataset* 3

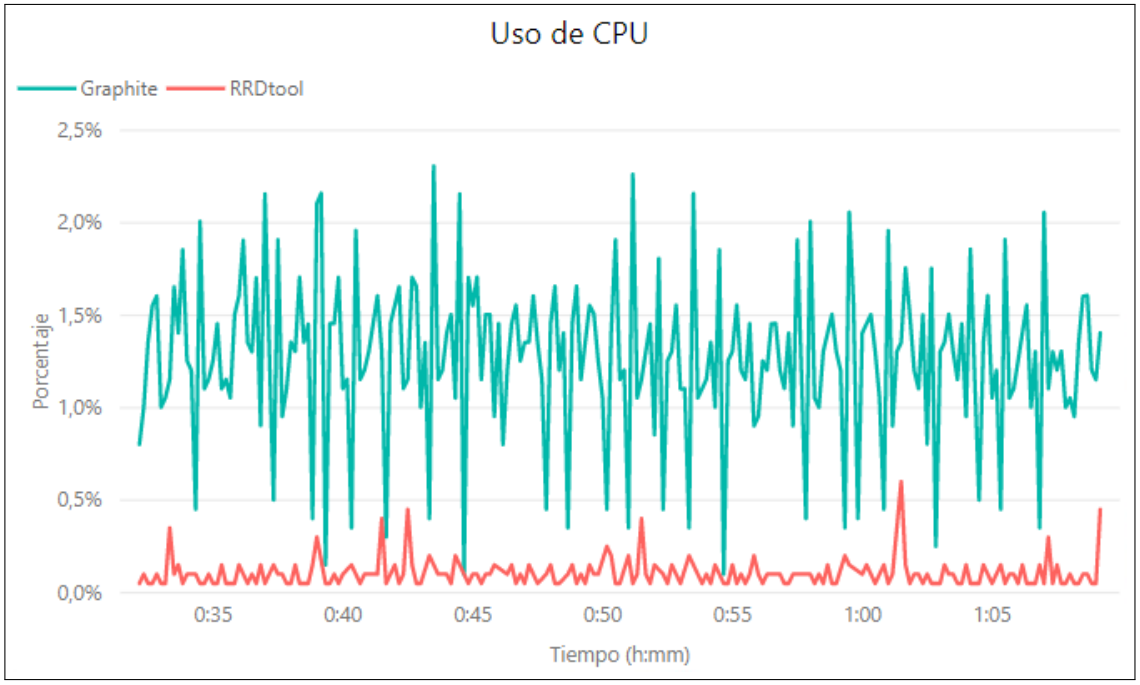
TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	30,45 %	979,24 MB	1,29 s	130,14 %	1,44
Graphite	1,99 %	**729,92 MB**	**44,69 ms**	**24,86 %**	**0,41**
RRDtool	**0,23 %**	814,57 MB	111,82 ms	38,14 %	0,53

Cuadro 4.7: Resultados promedios de las métricas medidas al ejecutar el primer escenario del *dataset* 3 (elaboración propia).

A continuación, se realiza el *test* para el segundo escenario del *dataset* 3. Las figuras 4.36, 4.37 y 4.38 muestran el uso de la CPU, uso de la memoria RAM y tiempo de ejecución, respectivamente; mientras que las figuras 4.39 y 4.40 presentan la precisión mediante las métricas MAPE y MASE. Todos estos resultados evidencian un comportamiento similar a lo presentado en el escenario anterior, aunque cuentan con un aumento en sus valores.



(a)



(b)

Figura 4.36: (a) Uso de la CPU en función al tiempo de ejecución del segundo escenario presente en el *dataset* 3 para los tres motores de bases de datos. (b) Uso de la CPU en función al tiempo de ejecución del segundo escenario presente en el *dataset* 3, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

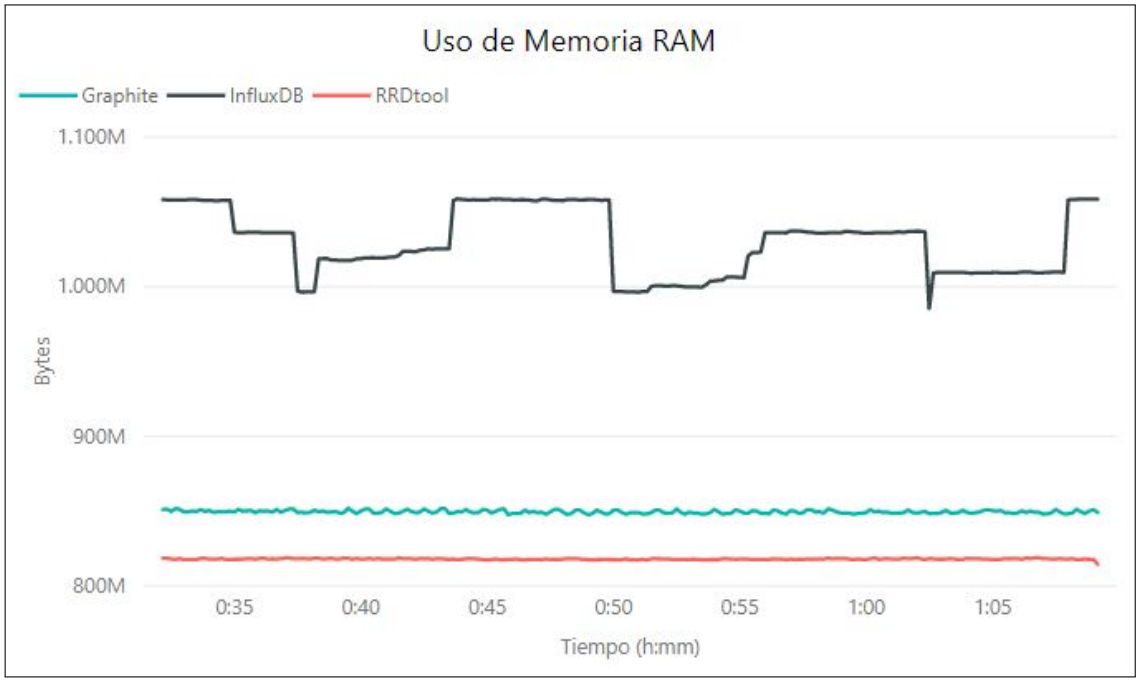
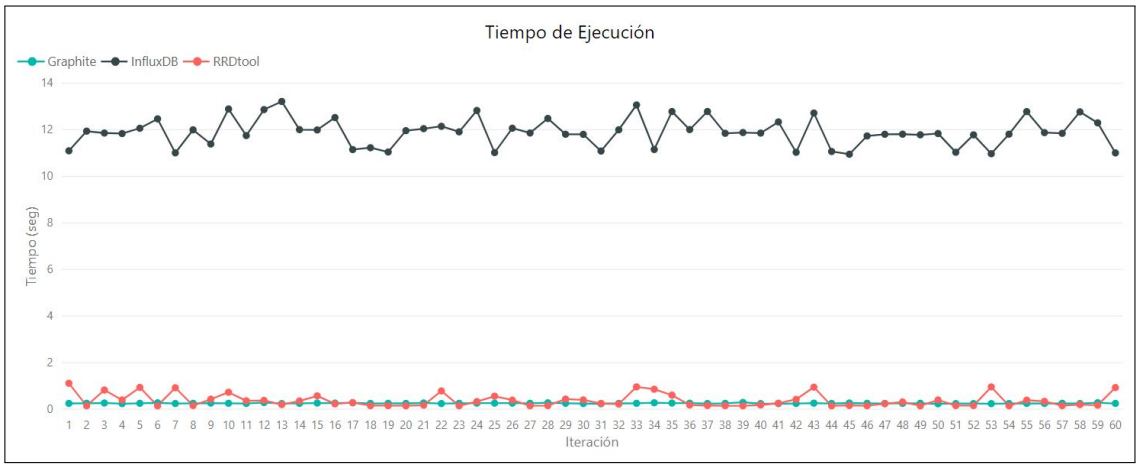
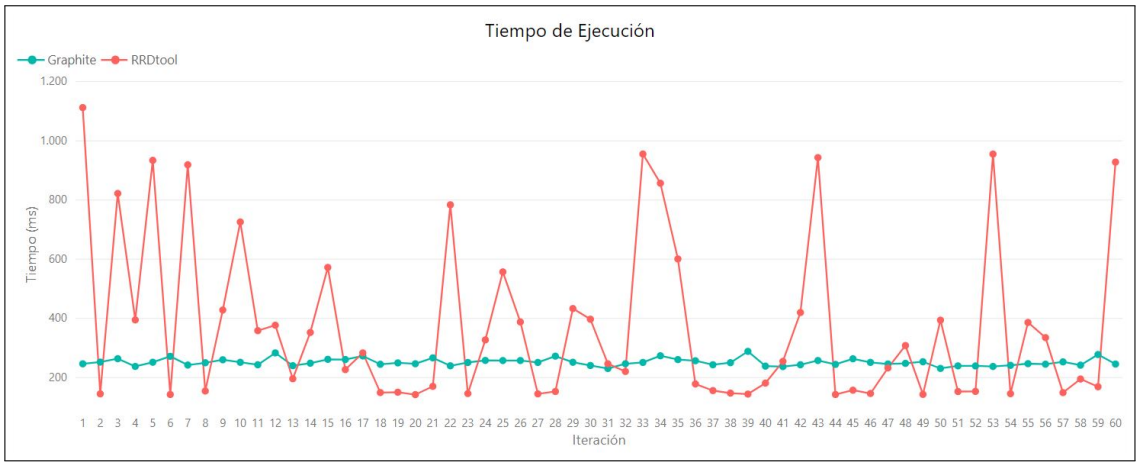


Figura 4.37: Uso de memoria RAM en función al tiempo de ejecución del segundo escenario presente en el *dataset* 3 (elaboración propia).



(a)



(b)

Figura 4.38: (a) Tiempo de ejecución de cada iteración correspondiente al segundo escenario del *dataset* 3 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al segundo escenario del *dataset* 3, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

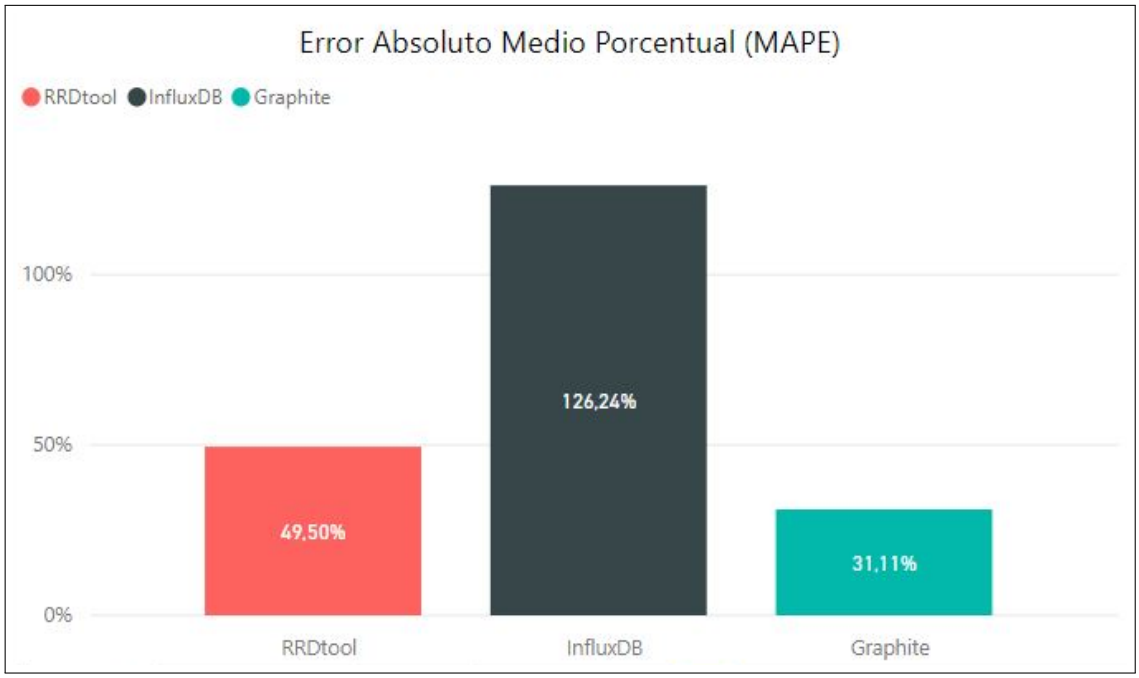


Figura 4.39: Promedio del MAPE correspondiente al segundo escenario del *dataset* 3 (elaboración propia).

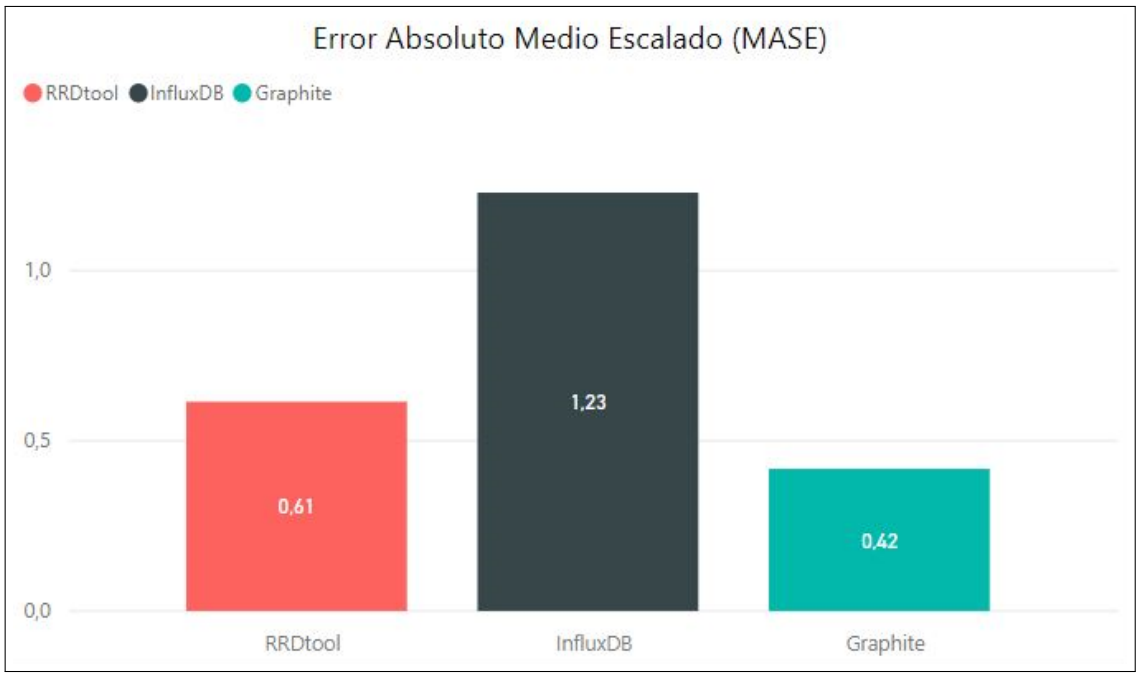


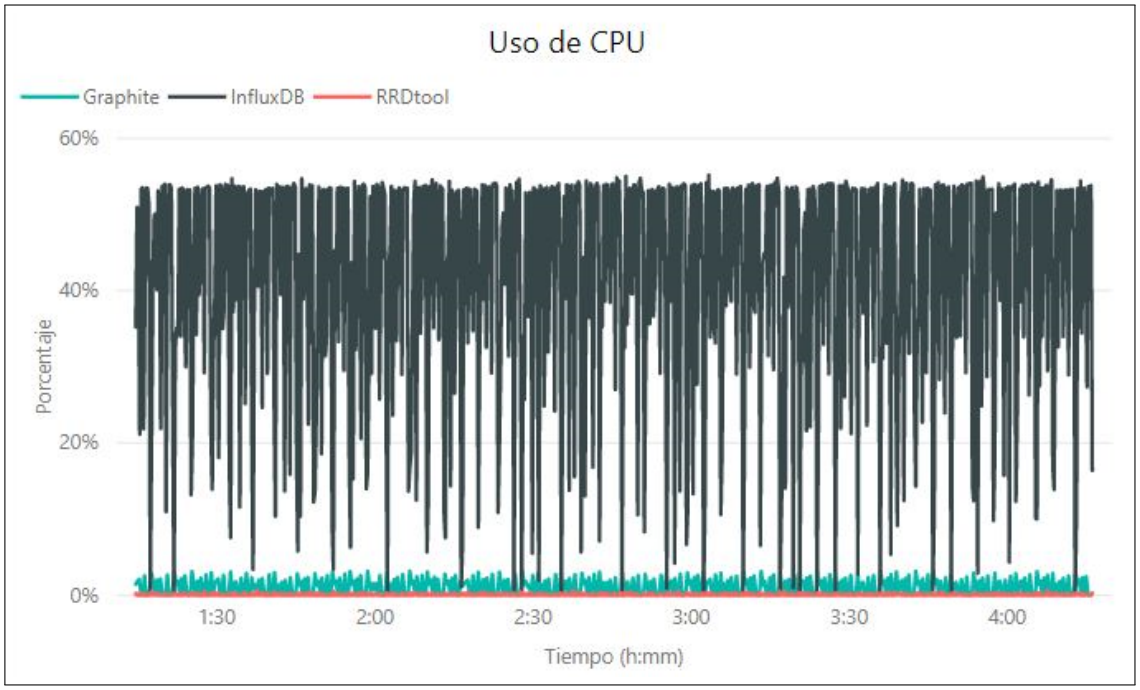
Figura 4.40: Promedio del MASE correspondiente al segundo escenario del *dataset* 3 (elaboración propia).

Resumen de resultados en el segundo escenario del *dataset* 3

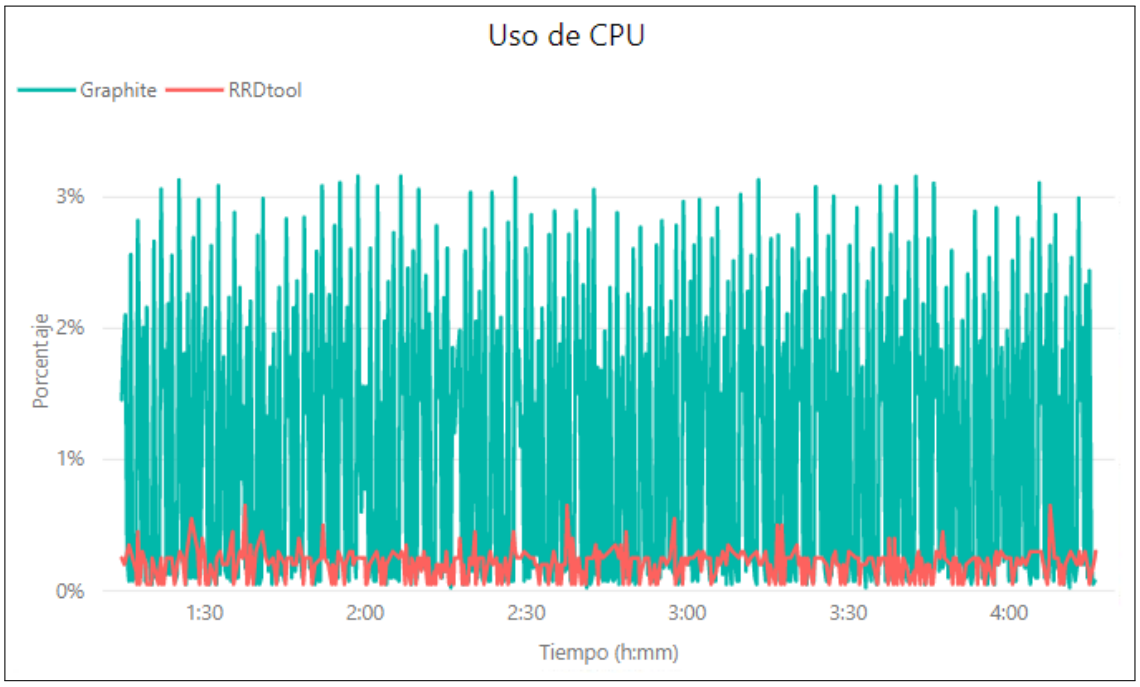
TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	40,92 %	1,03 GB	11,90 s	126,24 %	1,23
Graphite	1,27 %	849,56 MB	**251,98 ms**	**31,11 %**	**0,42**
RRDtool	**0,10 %**	**818,02 MB**	374,90 ms	49,50 %	0,61

Cuadro 4.8: Resultados promedios de las métricas medidas al ejecutar el segundo escenario del *dataset* 3 (elaboración propia).

Después se ejecuta el último *test* del experimento, correspondiente al tercer escenario del *dataset* 3. La figura 4.41 muestra un comportamiento similar a lo ya registrado anteriormente, aunque con un aumento considerable en la oscilación respecto al uso de la CPU en los tres motores de bases de datos. El uso de la memoria RAM (ver figura 4.42) mantiene la tendencia de escenarios anteriores aun cuando se elevan sus valores. La figura 4.43 presenta una notoria diferencia en el tiempo de ejecución de *InfluxDB* respecto a los otros dos motores. Las figuras 4.44 y 4.45 dan cuenta de un aumento en sus valores en las mediciones de error MAPE y MASE, aunque mantienen un comportamiento en su tendencia similar a los anteriores.



(a)



(b)

Figura 4.41: (a) Uso de la CPU en función al tiempo de ejecución del tercer escenario presente en el *dataset* 3 para los tres motores de bases de datos. (b) Uso de la CPU en función al tiempo de ejecución del tercer escenario en el *dataset* 3, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

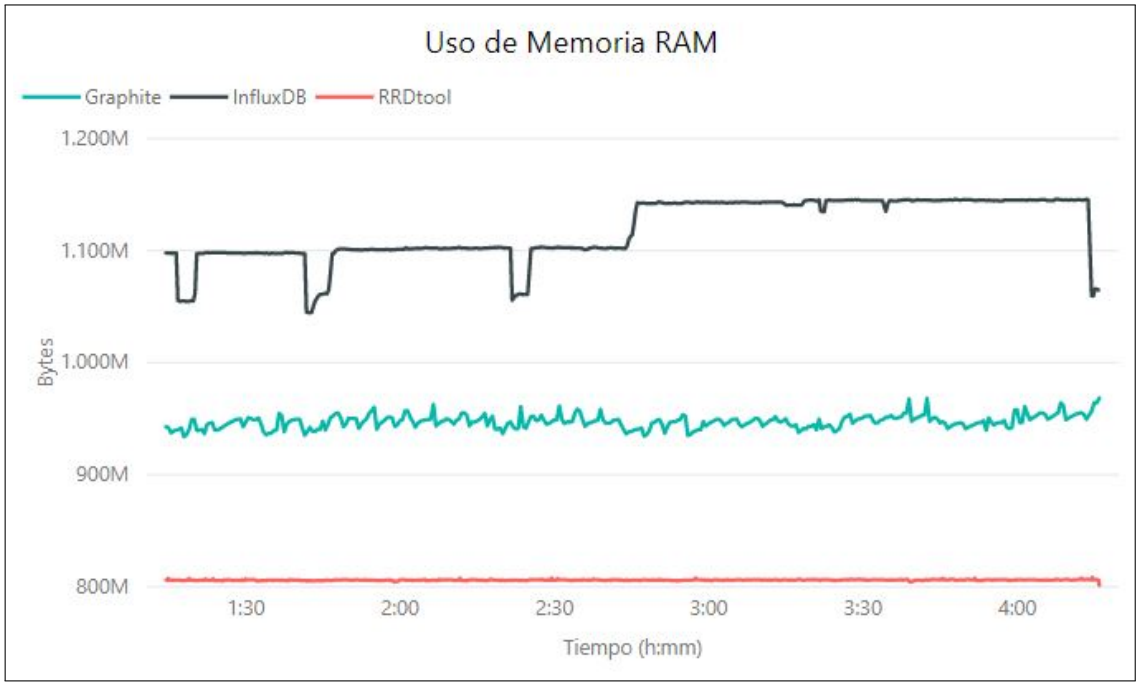
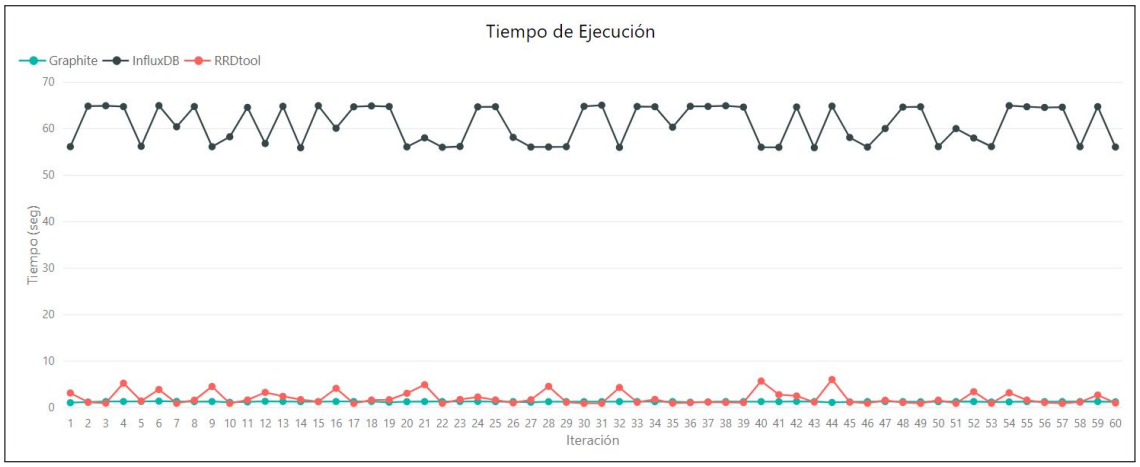
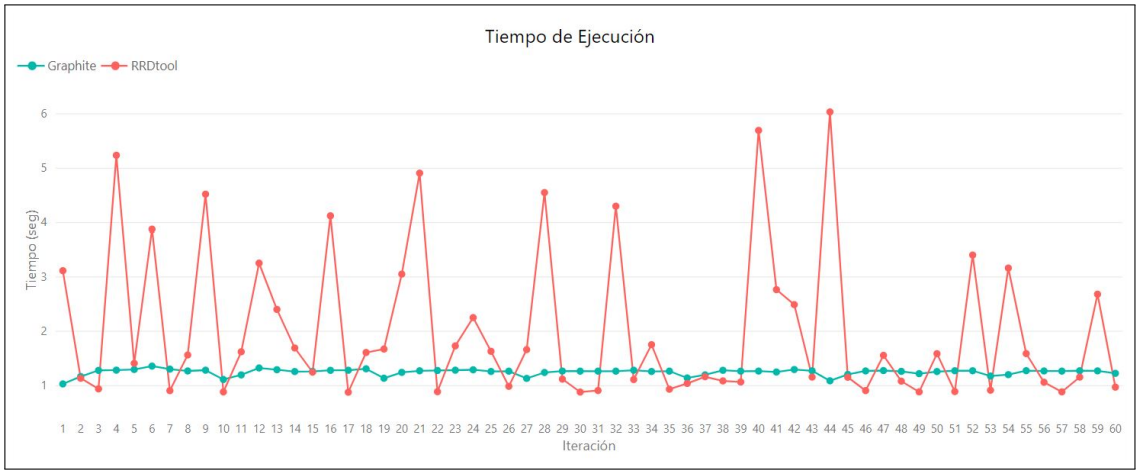


Figura 4.42: Uso de memoria RAM en función al tiempo de ejecución del tercer escenario presente en el *dataset* 3 (elaboración propia).



(a)



(b)

Figura 4.43: (a) Tiempo de ejecución de cada iteración correspondiente al tercer escenario del *dataset* 3 para los tres motores de bases de datos. (b) Tiempo de ejecución de cada iteración correspondiente al tercer escenario del *dataset* 3, sólo para los motores de bases de datos *Graphite* y *RRDtool* (elaboración propia).

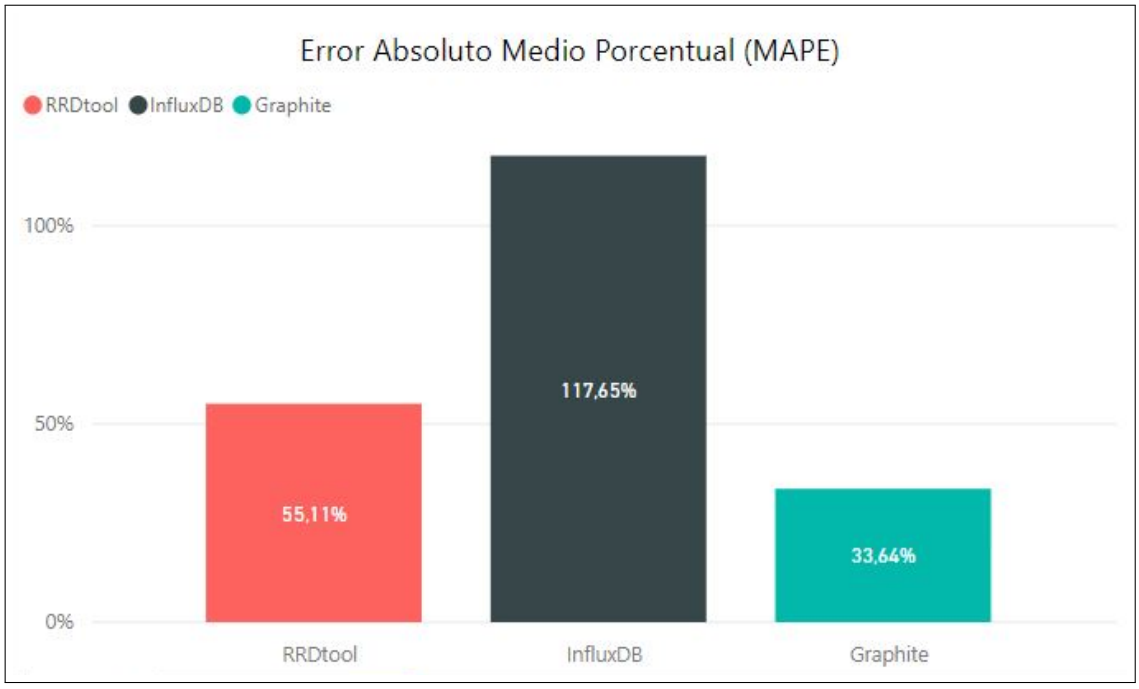


Figura 4.44: Promedio del MAPE correspondiente al tercer escenario del *dataset* 3 (elaboración propia).

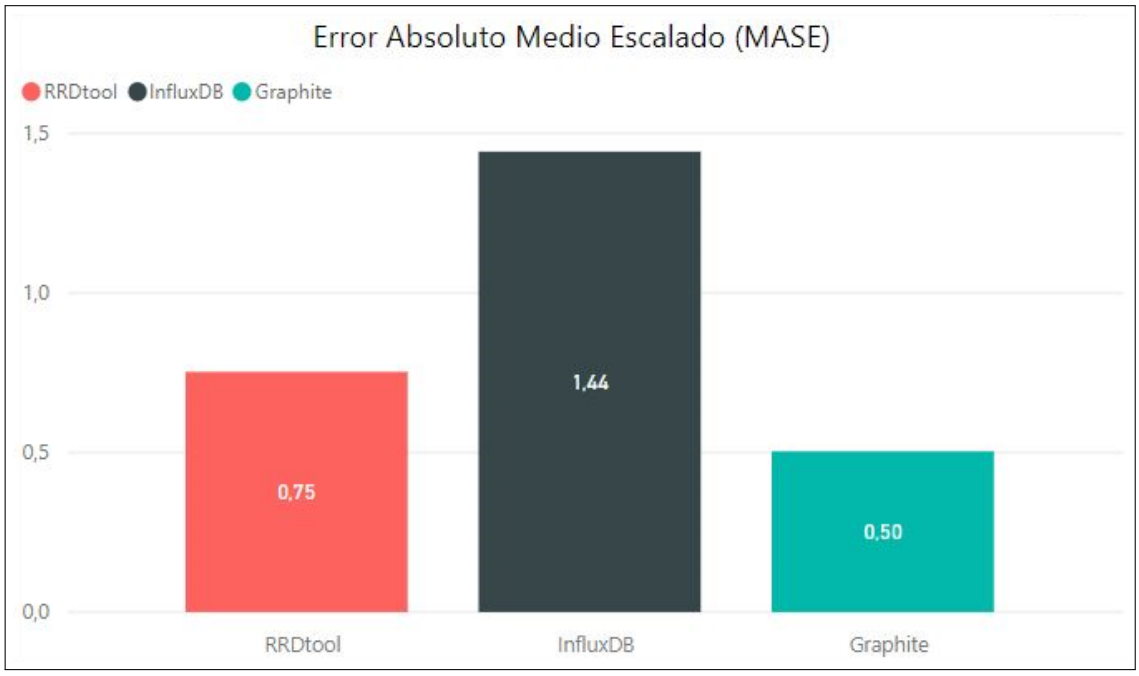


Figura 4.45: Promedio del MASE correspondiente al tercer escenario del *dataset* 3 (elaboración propia).

Resumen de resultados en el tercer escenario del *dataset* 3

TSDB	Uso de CPU	Uso de memoria RAM	Tiempo de ejecución	MAPE	MASE
InfluxDB	41,47 %	1,12 GB	60,97 s	117,65 %	1,44
Graphite	1,07 %	947,25 MB	**1,25 s**	**33,64 %**	**0,50**
RRDtool	**0,20 %**	**806,02 MB**	2,02 s	55,11 %	0,75

Cuadro 4.9: Resultados promedios de las métricas medidas al ejecutar el tercer escenario del *dataset* 3 (elaboración propia).

4.4. Análisis de los Resultados

Tras el desarrollo de la experimentación, se da a conocer que el motor de base de datos que mayor uso de la CPU tuvo a lo largo de los *tests* ejecutados corresponde a *InfluxDB*, el cual, a medida que se solicitaba una mayor cantidad de predicción en las consultas, requería de un mayor uso de la CPU. Sin embargo, a pesar de que dicha situación también ocurría tanto para *Graphite* como *RRDtool*, el aumento en el uso de la CPU resulta ser más significativo en *InfluxDB*.

El uso de memoria RAM corresponde a otra de las métricas de rendimiento que fue en aumento ante la solicitud de un mayor rango de predicción, siendo *InfluxDB* el motor que a lo largo de la experimentación demandó de un mayor consumo de memoria RAM. Por otro lado, *RRDtool* presentó el menor uso de memoria RAM, siendo igualado o superado por *Graphite* en algunos casos en que se solicitaba un corto rango de predicción. Una de las causas que explican las diferencias entre los tres motores de bases de datos corresponde a que *InfluxDB* carga una copia del *set* de datos completo en memoria, lo que ocasiona el alto consumo de memoria, por sobre los demás motores. Por otra parte, el comportamiento similar que presentan *Graphite* y *RRDtool* es debido a que la implementación de la función Holt-Winters en *RRDtool* fue portada a *Graphite* [45].

Otra de las métricas consideradas en el rendimiento es el tiempo de ejecución, en el cual los

motores *Graphite* y *RRDtool* presentan una baja oscilación respecto a sus tiempos promedios de ejecución, dando a conocer la estabilidad que sus consultas predictivas cuentan con los diferentes parámetros utilizados en la experimentación. En cambio, *InfluxDB* presenta un amplio rango de diferencia respecto a su tiempo de ejecución promedio, lo cual puede ser producto a que la configuración utilizada en la consulta no corresponde a la óptima, ocasionando que afecte el rendimiento de dicha consulta. Por otro lado, la causa en la alza en el tiempo de ejecución de *InfluxDB*, por sobre *Graphite* y *RRDtool*, se debe al formato de almacenamiento, el cual optimiza la escritura pero dificulta la consulta de estos datos [35].

Por otra parte, mediante las métricas MAPE y MASE se dan a conocer la precisión de los resultados en cada uno de los *tests* realizados. Se evidencia una mejor precisión por parte de *Graphite* y *RRDtool* dados los bajos valores de MAPE y MASE, por lo tanto, los modelos predictivos hechos por dichos motores son representativos al *dataset* en estudio. En cambio, *InfluxDB* presenta valores más elevados en ambas mediciones de error, lo que causa que los pronósticos en diferentes casos no sean un buen modelo para representar el comportamiento del *set* de datos.

Este último es más significativo con una mayor granularidad en sus datos y cuyo rango de predicción es mayor. Esto se debe a que la consulta predictiva de los motores de bases de datos (*Graphite* y *RRDtool*), entrega un resultado con una misma granularidad al *set* de datos en estudio, a diferencia del pronóstico de *InfluxDB* que es entregado en base a tendencias de la predicción realizada; así, la granularidad de los resultados de la consulta es menor en comparación a los demás motores, por lo que, su precisión disminuye.

4.5. Conclusiones del Experimento

Tras el desarrollo de la experimentación, se da a conocer algunos criterios en que destaca un motor de base de datos de series de tiempo por sobre otros. Es así como si se busca precisión en modelo predictivo, *RRDtool* ha sido el motor que ha mostrado mejor resultado. Sin embargo, y dado que los resultados obtenidos por *Graphite* no tienen una diferencia significativa respecto a *RRDtool*, ambos motores son recomendados al considerar la precisión

como prioridad.

Por otra parte, las herramientas con los mejores tiempos de ejecución fueron *Graphite* y *RRDtool*, siendo recomendadas para situaciones en que se requiera una alta frecuencia de ejecución de consultas predictivas, o bien si se desea automatizar estas para una constante ejecución de dichas consultas. En cambio, *InfluxDB* cuenta con un alto tiempo de ejecución, y su configuración dificulta su constante ejecución; es por esto que *InfluxDB* se recomienda para una situación opuesta, donde se requiere una baja frecuencia de ejecución a modo de permitir una configuración manual de sus parámetros. Además, la flexibilidad de este motor de base de datos para la modificación de los parámetros en las consultas predictivas, y su sintaxis similar a consultas SQL, permite que el manejo de las mismas sea de una manera más intuitiva respecto de la sintaxis de las consultas de los otros motores de base de datos de series de tiempo.

Los parámetros que tienen *InfluxDB* y *Graphite* permiten un rápido análisis si se desea modificar el escenario predictivo para obtener un buen modelo de pronóstico. En contraparte, *RRDtool* se centra en la predicción constante de un escenario específico de predicción, dado que la consulta predictiva está asociada al archivo donde se encuentran almacenados los datos, lo que limita en gran medida la modificación de los parámetros de dicha consulta.

Finalmente, *InfluxDB* y *Graphite* disponen de interfaces y *plugins* que permiten el uso de sus consultas de manera remota. Por otro lado, *RRDtool* es recomendado en escenarios donde las consultas se realizan de manera local, debido a que no cuenta con interfaces que faciliten su uso remoto, por lo que utilizarlo de dicha manera puede ser contraproducente en su rendimiento.

Conclusiones

En el desarrollo de esta memoria se abordó la comparativa de la precisión y rendimiento en bases de datos de series de tiempo, análisis que cobra importancia dado el interés que ha presentado este tipo de bases de datos en los últimos años, lo que hace relevante conocer más en profundidad las opciones y alternativas que presenta. Para esto, se desarrolló un *script* propio, utilizando como base el modelo de *benchmark* que realizan los *software* de *benchmarking* ya existentes, para efectuar la comparación con tres *datasets* diferentes, representando las posibilidades en que se pueden encontrar los datos en la práctica, considerando tres escenarios distintos relacionados al tamaño del rango a predecir.

El desarrollo de la memoria resultó ser una experiencia enriquecedora, donde se pudo aprender de las características y diferencias de las bases de datos de series de tiempo *InfluxDB*, *Graphite* y *RRDtool*, asimilando un nuevo tipo de funciones como las utilizadas en consultas predictivas; además de comprender el diseño de un modelo de *benchmarking* para sistemas informáticos, pudiendo desarrollar uno propio. Por otra parte, se debieron enfrentar inconvenientes, como la escasa documentación existente en las bases de datos *Graphite* y *RRDtool* relacionadas con las funciones predictivas que disponen, o bien, las configuraciones principales de los *plugins* utilizados en la herramienta *collectd*; los que dificultaron y demoraron una correcta implementación.

Se desarrolló un análisis comparativo tanto en precisión como en rendimiento en tres bases de datos de series de tiempo, centrándose en el uso de consultas predictivas. Se desarrolló e implementó un *script* que permite realizar el *benchmark* para poder comparar los motores de bases de datos bajo un mismo escenario. Para el desarrollo del proceso comparativo se utilizó un modelo de *benchmarking* para sistemas informáticos, aplicado en estudios similares,

donde dicho modelo fue adaptado para ajustarse al objetivo de este trabajo, incorporando nuevas secciones relacionadas con el pronóstico y precisión de ésta.

En el desarrollo de la memoria se logró cumplir con el objetivo general, el cual correspondía a realizar un estudio que permita discernir en primera instancia, entre un motor u otro según sea la perspectiva de interés, ya sea por precisión de la predicción o tiempo de ejecución de las consultas, apuntando a un escenario automatizado, siendo *RRDtool* y *Graphite* recomendadas para dicho escenario. Por otra parte, la flexibilidad en la configuración de los pronósticos y la gestión remota de estas son enfoques en los que *InfluxDB* y *Graphite* resultan ser una buena opción. Estos criterios pueden ser utilizados como punto de inicio en la recomendación de uso de un motor de bases de datos de series de tiempo ante proyectos que ameriten el uso de este tipo de bases de datos. Además, puede ser considerado como un punto de referencia ante estudios que involucren la ejecución de consultas predictivas en bases de datos de series de tiempo.

La labor que resultó ser más compleja de lo esperado fue la comprensión del funcionamiento de las bases de datos de series de tiempo y las consultas predictivas, las cuales presentaban una metodología diferente en cada motor de bases de datos, contando con elementos que requerían de un mayor tiempo de análisis e investigación para su comprensión, como es el caso de las políticas de retención presentes en los motores de bases de datos de series de tiempo, el cual determina el período y granularidad con que se guardarán los datos, controlando así el tamaño ocupado por los datos; siendo de gran importancia en la decisión de cómo se almacenan los *datasets* en las bases de datos.

Otra labor que resultó ser compleja fue la implementación de un *script* adecuado a lo que se necesitaba para realizar el *benchmark* de las consultas predictivas. Esto debido a que los *benchmarks* ya existentes tienen un enfoque a otro tipo de consultas, como lo son las consultas de agregación, que demandan de un análisis diferente a las consultas utilizadas en el desarrollo de esta memoria, centrándose en el rendimiento de estas. Las consultas predictivas requieren además de dicha comparación, en que se incluyen métricas como uso de la CPU, uso de memoria RAM y tiempo de ejecución; un contraste en la precisión de sus resultados, integrando métricas como MAPE y MASE para un análisis más detallado de los datos obtenidos en los pronósticos. Siendo la igualdad del entorno un factor protagónico en

la implementación para que se aborde desde un punto imparcial y objetivo durante el estudio.

Como trabajo futuro se propone la identificación de criterios para evaluar alternativas de bases de datos de series de tiempo en consultas predictivas utilizando *cluster* de nodos; esto dado que es una arquitectura que usualmente se encuentra en la práctica, pero debido a la dificultad que se presentó durante el desarrollo de la memoria en el que para implementar una arquitectura de multinodos en *InfluxDB*, que permita la utilización de dicho motor sin ningún tipo de limitación, requiere de su versión empresarial, y dado su costo actual de licenciamiento, esta resultó ser un impedimento para realizar el análisis para una arquitectura con multinodos. Sin embargo, hoy en día se están dando mayores posibilidades para acceder a dicha versión, puesto que *influxdata* anunció la posibilidad de utilizar la versión empresarial durante un período de prueba, siendo una buena oportunidad para la exploración del motor *InfluxDB* al tener en disposición todas sus capacidades.

Por otra parte, sería interesante el estudio comparativo que permita contrastar el rendimiento y precisión que presentan los pronósticos realizados mediante el uso de consultas predictivas con aquella que emplea un *software* estadístico, para así poder determinar las ventajas y desventajas en el uso de cada una de ellas, permitiendo reconocer la viabilidad de uso de una de estas formas de predicción por sobre la otra, o bien si estas se complementan para un mejor resultado en el análisis del comportamiento de los datos.

Finalmente, el desarrollo de la memoria es un gran aporte al desarrollo profesional, puesto que constituye un primer acercamiento al uso de un modelo de *benchmark* para sistemas informáticos y de las bases de datos de series de tiempo en conjunto a las consultas predictivas, de las cuales no se contaba con conocimientos previos, lo que ameritó aprender y profundizar respecto a nuevos temas, conociendo el funcionamiento de las bases de datos de series de tiempo y el cómo ejecutar consultas predictivas en ellas para luego integrar ambas en un estudio comparativo empleando un modelo de *benchmarking*.

La formación profesional entregada por la universidad constituye un pilar fundamental de conocimiento para el desarrollo de este trabajo, donde se destaca asignaturas electivas como Bases de Datos Avanzadas, en la cual se estudiaron diferentes tipos de bases de datos con el objetivo de cambiar el enfoque tradicional al que uno está acostumbrado con el frecuente

uso de las bases de datos relacionales, permitiendo que la comprensión de un nuevo tipo de base de datos sea más intuitivo. Otra de las asignaturas electivas fue Bases Tecnológicas para la Inteligencia de Negocios, siendo de gran apoyo en este trabajo en la selección y manejo de la información obtenida con los resultados de la experimentación, donde se aprovechó los conocimientos adquiridos en el uso de las herramientas vistas en trabajos realizados en dicha asignatura. Ambas asignaturas mencionadas representan la base en el interés por profundizar sobre esta área informática.

Bibliografía

[1] Diccionario de la Real Academia Española. [en línea] <www.rae.es>, 2018. [consulta: 8 enero 2018].

[2] Oxford Dictionaries. [en línea] <https://www.oxforddictionaries.com/>, 2018. [consulta: 16 abril 2018].

[3] Andreas Bader. *Comparison of Time Series Databases*. University of Stuttgart, Enero 2016.

[4] Anthony G. Barnston. *Correspondence among the Correlation, RMSE, and Heidke Forecast Verification Measures; Refinement of the Heidke Score*. Climate Analysis Center, 1992.

[5] Apache Software Foundation. Apache ZooKeeper. [en línea] <https://zookeeper.apache.org/>. [consulta: 10 febrero 2018].

[6] Australian Bureau of Statistics. Time Series Analysis: The Basics. [en línea] <http://www.abs.gov.au/websitedbs/D3310114.nsf/home/Time+Series+Analysis:+The+Basics>. [consulta: 10 enero 2018].

[7] Box, G.E.P. y Kenkins, G.M. *Time Series Analysis, Forecasting and Control*. Editorial Holden-Day, San Francisco, 2da Edición, 1976.

[8] Brian F. Cooper. YCSB-TS. [en línea] <https://github.com/TSDBBench/YCSB-TS>. [consulta: 25 enero 2018].

[9] Cairo. Cairo. [en línea] <https://cairographics.org/>. [consulta: 10 febrero 2018].

[10] Chai, Tianfeng and R. Draxler, R. *Root mean square error (RMSE) or mean absolute error (MAE)?*, volume 7. Geosci. Model Dev., 01 2014.

[11] Collectd. Collectd – The system statistics collection daemon. [en línea] <https://collectd.org/>. [consulta: 18 mayo 2018].

[12] David Gerbing. *Time Series Components*. Portland State University, Enero 2016.

[13] Django. Django. [en línea] <https://www.djangoproject.com/>. [consulta: 10 febrero 2018].

[14] Druid. Druid — Interactive Analytics at Scale. [en línea] <http://druid.io/>. [consulta: 10 febrero 2018].

[15] D.S. Kaippallimalil J. Jacob. FinTime - a financial time series benchmark. [en línea] <https://cs.nyu.edu/shasha/fintime.html>. [consulta: 25 enero 2018].

[16] Alice Bérard Georges Hébrail. Individual household electric power consumption data set. [en línea] <https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption>, 2012. [consulta: 15 marzo 2017].

[17] Grafana Labs. Grafana - The open platform for analytics and monitoring. [en línea] <https://grafana.com/>. [consulta: 18 mayo 2018].

[18] Graphite. Functions. [en línea] <http://graphite.readthedocs.io/en/latest/functions.html>. [consulta: 1 febrero 2018].

[19] Graphite. Tools That Work With Graphite. [en línea] <https://graphite.readthedocs.io/en/latest/tools.html>. [consulta: 1 febrero 2018].

[20] Graphite. Graphite. [en línea] <http://graphiteapp.org/>, 2017. [consulta: 29 junio 2016].

[21] HAProxy. HAProxy. [en línea] <http://www.haproxy.org/>. [consulta: 10 febrero 2018].

[22] Hayato Matsuura. Grafana RRD Server. [en línea] <https://github.com/doublemarket/grafana-rrd-server>. [consulta: 18 mayo 2018].

[23] Hyndman, R.J. and Athanasopoulos, G. Autoregressive models. [en línea] <https://www.otexts.org/fpp/8/3>, 2013. Sección 8/3. [consulta: 5 febrero 2018].

[24] Hyndman, R.J. and Athanasopoulos, G. Backshift notation. [en línea] <https://www.otexts.org/fpp/8/2>, 2013. Sección 8/2. [consulta: 5 febrero 2018].

[25] Hyndman, R.J. and Athanasopoulos, G. Forecasting: principles and practice. [en línea] <https://www.otexts.org/fpp/7>, 2013. Sección 7. [consulta: 29 junio 2016].

[26] Hyndman, R.J. and Athanasopoulos, G. Forecasting: principles and practice. [en línea] <https://www.otexts.org/fpp/7/2>, 2013. Sección 7/2. [consulta: 30 enero 2018].

[27] Hyndman, R.J. and Athanasopoulos, G. Forecasting: principles and practice. [en línea] <https://www.otexts.org/fpp/7/5>, 2013. Sección 7/2. [consulta: 30 enero 2018].

[28] Hyndman, R.J. and Athanasopoulos, G. Forecasting: principles and practice. [en línea] <https://www.otexts.org/fpp/2/5>, 2013. Sección 2/5. [consulta: 10 enero 2018].

[29] Hyndman, R.J. and Athanasopoulos, G. Moving average models. [en línea] <https://www.otexts.org/fpp/8/4>, 2013. Sección 8/4. [consulta: 5 febrero 2018].

[30] Hyndman, R.J. and Athanasopoulos, G. Non-seasonal ARIMA models. [en línea] <https://www.otexts.org/fpp/8/5>, 2013. Sección 8/5. [consulta: 5 febrero 2018].

[31] Hyndman, R.J. and Athanasopoulos, G. Seasonal ARIMA models. [en línea] <https://www.otexts.org/fpp/8/9>, 2013. Sección 8/9. [consulta: 5 febrero 2018].

[32] Hyndman, R.J. and Athanasopoulos, G. Stationarity and differencing. [en línea] <https://www.otexts.org/fpp/8/1>, 2013. Sección 8/1. [consulta: 5 febrero 2018].

[33] Influxdata. InfluxDB-Python. [en línea] <http://influxdb-python.readthedocs.io/en/latest/include-readme.html>. [consulta: 1 febrero 2018].

[34] Influxdata. Single Node or Cluster? [en línea] <https://docs.influxdata.com/influxdb/v1.4/guides/hardware_sizing/#single-node-or-cluster>, 2017. [consulta: 5 diciembre 2017].

[35] Influxdata. Storage engine and the Time-Structured Merge Tree (TSM). [en línea] <https://docs.influxdata.com/influxdb/v1.5/concepts/storage_engine/>, 2017. [consulta: 15 junio 2018].

[36] Influxdata. API client libraries. [en línea] <https://docs.influxdata.com/influxdb/v1.4/tools/api_client_libraries/>, 2018. [consulta: 1 febrero 2018].

[37] Influxdata. Clustering. [en línea] <https://docs.influxdata.com/influxdb/v1.4/high_availability/clusters/>, 2018. [consulta: 1 febrero 2018].

[38] Influxdata. InfluxDB. [en línea] <https://www.influxdata.com/time-series-platform/influxdb/>, 2018. [consulta: 1 febrero 2018].

[39] Influxdata. InfluxQL functions. [en línea] <https://docs.influxdata.com/influxdb/v1.4/query_language/functions>, 2018. [consulta: 1 febrero 2018].

[40] Influxdata. JSON Protocol. [en línea] <https://docs.influxdata.com/influxdb/v0.9/write_protocols/json/>, 2018. [consulta: 1 febrero 2018].

[41] Influxdata. Line Protocol tutorial. [en línea] <https://docs.influxdata.com/influxdb/v1.4/write_protocols/line_protocol_tutorial/#cli>, 2018. [consulta: 1 febrero 2018].

[42] Influxdata. Supported Protocols. [en línea] <https://docs.influxdata.com/influxdb/v1.4/supported_protocols/>, 2018. [consulta: 1 febrero 2018].

[43] Influxdata. Writing data with the HTTP API. [en línea] <https://docs.influxdata.com/influxdb/v1.4/guides/writing_data/>, 2018. [consulta: 1 febrero 2018].

[44] Janet Wesner. MAE and RMSE - Wich Metric is Better? [en línea] <https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d>, Marzo 2016. [consulta: 10 enero 2018].

[45] Jason Dixon. *Monitoring with Graphite: Tracking Dynamic Host and Application Metrics at Scale*. O'Reilly Media, 2017.

[46] J.C. Ramesh Reddy, T. Ganesh, M. Venkateswaran, PRS Reddy. *Forecasting of Monthly Mean Rainfall in Coastal Andhra*. International Journal of Statistics and Applications, Vol. 7 No. 4, 2017.

[47] Jorge Galbiati Riesco. *Métodos Elementales de Procesamiento de Series de Tiempo*. 2012.

[48] Juan Julián Merelo Guervós. *Selección y Configuración de Sistemas Informáticos: Benchmarking*. Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada, 2003.

[49] KairosDB Team. KairosDB. [en línea] <http://kairosdb.github.io/>, 2015. [consulta: 10 febrero 2018].

[50] Kenneth Reitz. Requests: HTTP for Humans. [en línea] <http://python-requests.org>. [consulta: 18 mayo 2018].

[51] Sungil Kim and Heeyoung Kim. *A new metric of absolute percentage error for intermittent demand forecasts*, volume 32. International Journal of Forecasting, 2016.

[52] M. Lichman. UCI machine learning repository. [en línea] <http://archive.ics.uci.edu/ml>, 2013. [consulta: 15 marzo 2017].

[53] LinkedIn Corp. Databus. [en línea] <https://github.com/linkedin/databus>, 2015. [consulta: 10 febrero 2018].

[54] Minitab Express. ¿Qué es ANOVA? [en línea] <https://support.minitab.com/es-mx/minitab/18/help-and-how-to/modeling-statistics/anova/supporting-topics/basics/what-is-anova/>. [consulta: 18 mayo 2018].

[55] Noah Spurrier. Pexpect version 4.5. [en línea] <http://pexpect.readthedocs.io/en/stable/index.html>, 2013. [consulta: 18 mayo 2018].

[56] okITup. ¿Qué es InfluxDB y como funciona? [en línea] <https://www.okitup.com/blog/que-es-influxdb/>, abril 2015. [consulta: 29 junio 2016].

[57] Oliver Kopp. TSDBBench. [en línea] <https://tsdbbench.github.io/>. [consulta: 25 enero 2018].

[58] Oracle VM VirtualBox. Welcome to VirtualBox.org! [en línea] <https://www.virtualbox.org/>. [consulta: 18 mayo 2018].

[59] PennState Eberly College of Science. Decomposition Models. [en línea] <https://onlinecourses.science.psu.edu/stat510/node/69>. [consulta: 25 enero 2018].

[60] Pilar González Casimiro. *Análisis de Series Temporales: Modelos ARIMA*. Departamento de Economía Aplicada III. Universidad del País Vasco, 2009.

[61] Pranay Kumar Rahi, Rajesh Mehra. *Analysis of Power Spectrum Estimation Using Welch Method for Various Window Techniques*. International Journal of Emerging Technologies and Engineering (IJETE) ISSN: 2348–8050, Agosto 2014.

[62] Prometheus. Aggregation operators. [en línea] <https://prometheus.io/docs/prometheus/latest/querying/operators/#aggregation-operators>. [consulta: 1 febrero 2018].

[63] Prometheus. Client Libraries. [en línea] <https://prometheus.io/docs/instrumenting/clientlibs/>. [consulta: 1 febrero 2018].

[64] Prometheus. Prometheus. [en línea] <https://prometheus.io/>. [consulta: 1 febrero 2018].

[65] Prometheus. When to use the Pushgateway. [en línea] <https://prometheus.io/docs/practices/pushing/>. [consulta: 1 febrero 2018].

[66] Real Options Valuation. *Box-Jenkins ARIMA Advanced Time Series*. ROV Technical Papers Series: Volume 25, 2012.

[67] Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Cross-Validation*, pages 532–538. Encyclopedia of Database Systems. Springer US, Boston, 2009.

[68] Reguant-Álvarez, M. y Torrado-Fonseca, M. *El método Delphi*, volume 9 (1). REIRE, Revista d'Innovació i Recerca en Educació, 2016.

[69] Rob J. Hyndman. *Another look at Forecast-Accuracy Metrics for Intermittent Demand*. Monash University, Australia, Junio 2006.

[70] Rob J. Hyndman. Cross-validation for time series. [en línea] <https://robjhyndman.com/hyndsight/tscv/>, 2016. [consulta: 1 febrero 2018].

[71] Robust Perception. Robust Perception. [en línea] <https://www.robustperception.io/>. [consulta: 1 febrero 2018].

[72] Santiago de la Fuente Fernández. MODELO ARIMA(p,d,q)(P,D,Q)s. [en línea] <http://www.estadistica.net/ECONOMETRIA/SERIES-TEMPORALES/modelo-arima.pdf>. [consulta: 5 febrero 2018].

[73] solid IT. DB-Engines Ranking of Time Series DBMS. [en línea] <https://db-engines.com/en/ranking/time+series+dbms>, 2018. [consulta: 12 junio 2018].

[74] solid IT. DB-Engines Ranking per database model category. [en línea] <https://db-engines.com/en/ranking_categories>, 2018. [consulta: 10 diciembre 2017].

[75] solid IT. Method of calculating the scores of the DB-Engines Ranking. [en línea] <https://db-engines.com/en/ranking_definition>, 2018. [consulta: 29 junio 2017].

[76] solid IT. Time Series DBMS. [en línea] <https://db-engines.com/en/article/Time+Series+DBMS>, 2018. [consulta: 29 junio 2017].

[77] STAC Benchmark Council. STAC-M3 Benchmark Suite. [en línea] <https://stacresearch.com/sites/default/files/d5root/STAC-M3_Antuco_Overview.pdf>. [consulta: 25 enero 2018].

[78] Stanislav Sinyagin. RRDman Project Homepage. [en línea] <http://rrfw.sourceforge.net/rrdman/>. [consulta: 18 mayo 2018].

[79] T., ELIANA MIRLEDY and M., ALEXANDER and G., ALEJANDRO. *Pronóstico de bolsa de valores empleando técnicas inteligentes*, volume 9. Tecnura, 2006.

[80] T. Goldschmidt, A. Jansen, H. Koziolek, J. Doppelhamer, H. Breivold. *Scalability and Robustness of Time-Series Databases for Cloud-Native Monitoring of Industrial Processes*. IEEE 7th International Conference on Cloud Computing, Junio 2014.

[81] T. Wlodarczyk. *Overview of Time Series Storage and Processing in a Cloud Environment*. Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference, Diciembre 2012.

[82] TempoIQ. TempoIQ. [en línea] <https://www.tempoiq.com/>, 2016. [consulta: 10 febrero 2018].

[83] Tobias Oetiker. rrdcreate. [en línea] <https://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html>. [consulta: 1 febrero 2018].

[84] Tobias Oetiker. rrdtool. [en línea] <https://oss.oetiker.ch/rrdtool/doc/rrdtool.en.html>. [consulta: 1 febrero 2018].

[85] Tobias Oetiker. RRDtool Programming. [en línea] <https://oss.oetiker.ch/rrdtool/prog/index.en.html>. [consulta: 1 febrero 2018].

[86] Tobias Oetiker. What RRDtool does. [en línea] <https://oss.oetiker.ch/rrdtool/index.en.html>, febrero 2017. [consulta: 29 junio 2016].

[87] Travis CI. Aggregation. [en línea] <http://opentsdb.net/docs/build/html/user_guide/query/aggregators.html#available-aggregators>. [consulta: 1 febrero 2018].

[88] Travis CI. Clients. [en línea] <http://opentsdb.net/docs/build/html/resources.html#clients>. [consulta: 1 febrero 2018].

[89] Travis CI. What's New. [en línea] <http://opentsdb.net/docs/build/html/new.html#x-planned>. [consulta: 1 febrero 2018].

[90] Travis CI. The Scalable Time Series Database. [en línea] <http://opentsdb.net/index.html>, 2017. [consulta: 29 junio 2016].

[91] Artur Trindade. Electricityloaddiagrams20112014 data set. [en línea] <https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014>, 2015. [consulta: 15 marzo 2017].

[92] Twisted Matrix Labs. Twisted. [en línea] <https://twistedmatrix.com/trac/>. [consulta: 10 febrero 2018].

[93] Varnish Cache. Varnish HTTP Cache. [en línea] <https://varnish-cache.org/>. [consulta: 10 febrero 2018].

[94] Saverio De Vito. Air quality data set. [en línea] <https://archive.ics.uci.edu/ml/datasets/Air+quality>, 2016. [consulta: 15 marzo 2017].

[95] Wes McKinney. Python data analysis library. [en línea] <https://pandas.pydata.org/>, 2011. [consulta: 1 febrero 2018].

[96] Will Welch. Seasonal. [en línea] <https://github.com/welch/seasonal>. [consulta: 18 mayo 2018].

[97] Williams J. Stevenson. *Análisis de Regresión*. Estadística para Administración y Economía, 2008.