UNIVERSIDAD DE LA CIUDAD DE AGUASCALIENTES

MAESTRÍA EN CIENCIA DE DATOS

ANÁLISIS TOPOLÓGICO DE DATOS

“Detección de anomalías de tráfico en servidores web”

Alumno:

E23S-18014: MITSIU ALEJANDRO CARREÑO SARABIA

Periodo Enero 2024 - Junio 2024, Aguascalientes, Ags

1. Introducción

Se propone un método basado en modelos matemáticos y estadísticos para evaluar y detectar valores anómalos en las conexiones que recibe un servidor web. Mediante este método es posible evaluar nuevas peticiones basado en el tráfico histórico del servidor y obtener un índice de similitud respecto a solicitudes pasadas, con ello es posible detectar anomalías o contenido malicioso y tomar acciones tanto preventivas como correctivas.

2. Problemática

Con la expansión del acceso a servicios de internet, así como la creciente disponibilidad de dispositivos de distintas categorías para conectarse a la red, la demanda y tráfico de servicios web se encuentra en constante aumento. Mucho se ha desarrollado en términos de escalabilidad de infraestructura así como adopción de soluciones distribuidas para dar servicio a la creciente demanda[1], desde la producción en masa de dispositivos celulares y móviles, hasta la progresiva adopción del internet de las cosas, pero derivado de dicha disponibilidad, se genera una cantidad enorme de tráfico que cualquier servidor web disponible desde internet debe dar seguimiento, procesar y contestar[2].

El origen de dicho tráfico puede ser generado por usuarios reales, conexiones de bots, conexiones automatizadas y conexiones de usuarios con intenciones maliciosas cada uno de estos grupos de clientes se comportan de maneras muy específicas[3], solo analizar la cantidad de información generada de manera manual es una tarea imposible.

El tráfico de servidores web tiene claras tendencias como recursos solicitados, región geográfica de donde se solicita, hora en que se solicitó, cantidad de bytes enviados, por lo que identificar las tendencias y detectar las anomalías es un trabajo que puede ser automatizado y al que se le pueden aplicar distintas técnicas de análisis estadístico y topológico que permitan analizar la información desde múltiples dimensiones y perspectivas.

Como consecuencia del acceso generalizado a servicios de internet, se ha desarrollado una grán tendencia de alojar datos personales, identificables y sensibles[4] lo que aumenta la importancia de evaluar qué y cómo se están accediendo a los recursos solicitados, así como desarrollar herramientas que faciliten filtrar las anomalías para tomar acciones preventivas y correctivas[5].

3. Estado del arte

Para realizar el estudio se emplearon datos de un servidor web real el cual aloja múltiples dominios, los datos crudos corresponden a las bitácoras de conexión del software nginx, el cuál es empleado como proxy inverso.

Los datos empleados en el estudio comprenden un periodo de 15 días, a pesar de ello se lograron registrar más de un millón de conexiones, por cada conexión se guarda la siguiente información:

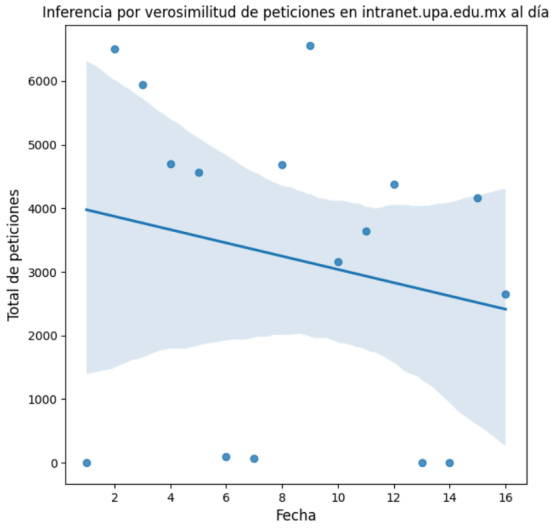
- ***remote_addr:*** Dirección IPv4 del cliente que inició la conexión.
- ***remote_usr:*** Nombre de usuario (solo aplica si el usuario está autenticado).
- ***date***: Fecha en formato “dd/MM/YYYY”.
- ***time:*** Hora en formato “HH:MM:SS”.
- ***request:*** Descripción del recurso solicitado por el cliente.
- ***req_method:*** Método HTTP mediante el cual se solicitó el recurso.
- ***req_uri:*** Dirección URI del recurso solicitado.
- ***http_ver:*** Versión HTTP bajo la que se estableció la conexión cliente-servidor.
- ***status:*** Código de estatus HTTP que resolvió el servidor.
- ***body_bytes_sent:*** Cantidad de bytes enviados en la respuesta.
- ***http_referer:*** Valor de la cabecera http_referer con el que se conectó el cliente.
- ***user_agent:*** Valor de la cabecera user_agent con el que se conectó el cliente.
- ***dec_req_uri:*** Es una réplica decodificada del valor req_uri.
- ***clean_path:*** Filtrado del valor req_uri únicamente con el recurso solicitado (sin parámetros).
- ***clean_query_list:*** Listado de python de los parámetros query.

- ***domain:*** Nombre del dominio listado en el campo http_referer.
- ***fabstime:*** Valor time transformada en decimal.
- ***weekday:*** Valor numérico correspondiente al día de la semana donde Lunes (1) y Domingo (7).

A los datos descritos se les aplicaron los siguientes estudios:

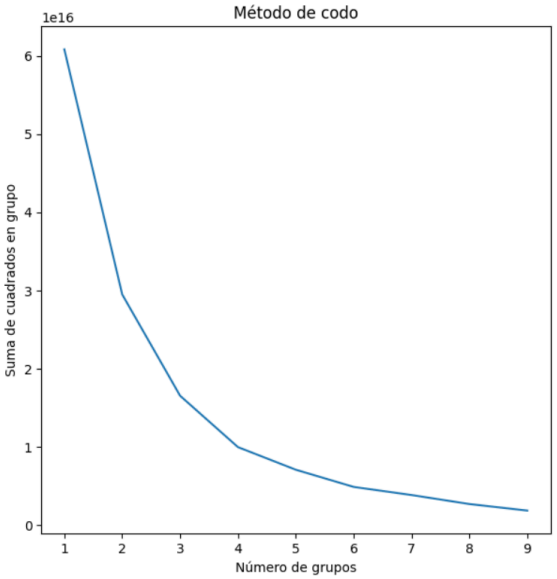
Regresión lineal: Segmentando únicamente a los datos del dominio "intranet.upa.edu.mx" se nota una clara tendencia respecto a las horas y días de mayor demanda a los recursos del servidor con principal actividad de lunes a viernes; por lo que es posible determinar que la actividad del servidor para este dominio está estrechamente relacionada con la actividad de la institución educativa.

Al contar con los datos de frecuencia y fecha, es posible realizar una regresión lineal para estimar la cantidad de tráfico que recibirá el servidor por parte de este dominio.

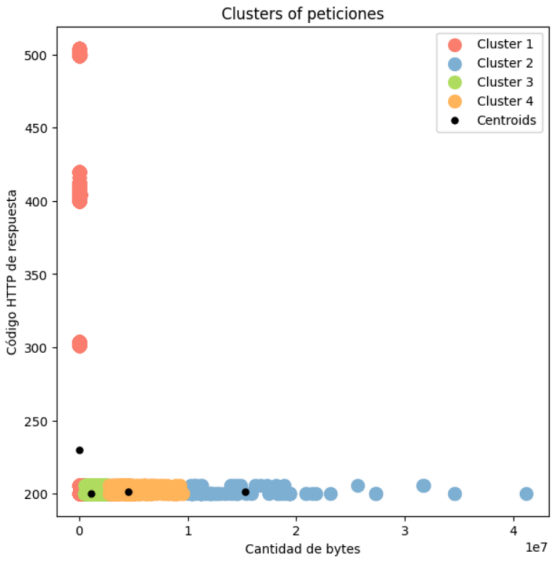


K-means: Empleando todos los datos del servidor se decidió agrupar las conexiones partiendo de la cantidad de bytes enviados y el estatus HTTP de la respuesta.

Mediante la heurística del "método del codo" para determinar la cantidad óptima de clusters basada en la varianza nos indica que la cantidad de clusters dado los datos es 4.



Una vez realizada la clusterización con los cuatro grupos se obtienen los siguientes resultados:



Se nota una clara tendencia a que las respuestas exitosas (2XX-3XX) contesten más información (cantidad de bytes) que una respuesta de error (4XX-5XX) lo cuál es esperado ya que una petición de error no debe responder más allá de informar del error mismo, esta es una posible técnica para identificar conexiones que regresen una cantidad anómala de bytes

Complejos simpliciales: Para la elaboración de los complejos simpliciales se comienza por graficar cada punto (renglón) con todas sus dimensiones (características) y estos se toman como base para generar los

0-simplices, es decir una nube de puntos multidimensional.

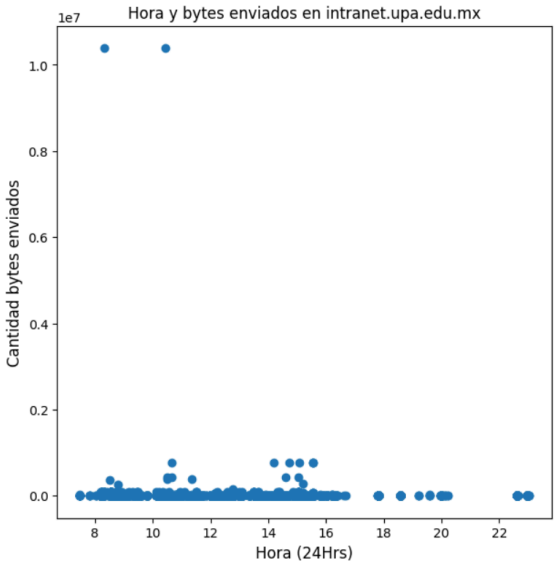
De estos puntos se propone conectar en 1-simplices (bordes) según el atributo “domain” el cual comprende subdominio (si existe) dominio y dominio de nivel superior (por ejemplo “stage.puntoderecarga.mx”) de esta manera se tiene una serie de 1-simplices (líneas) que conecta las conexiones según el subdominio y dominio al que apuntan.

A fin de generar los 2-simplices (triángulos) se puede tomar el atributo “status” el cuál hace referencia al código de estado HTTP que se respondió a cada petición en este punto se tendrá un conjunto de triángulos donde por cada triángulo corresponde al mismo dominio y subdominio y estatus.

Finalmente para generar los 3-simplices (tetraedros) es posible agregar el atributo “clean_path” el cuál contiene los directorios y subdirectorios (si existen) para llegar al recurso solicitado (por ejemplo “/intra/acceso_i.php”) cabe destacar que el atributo “clean_path” descarta cualquier parámetro o query de la url por lo que únicamente tiene la ruta y el recurso solicitado, de esta manera se controla la variabilidad de los parámetros.

Con la topología descrita se puede realizar un análisis de relación entre el código de estado HTTP y el recurso solicitado desagregado por cada uno de los sitios alojados, con ello es sencillo detectar solicitudes a recursos anómalos y evaluar cómo manejo el servidor dicha solicitud, también es posible detectar si el servidor está manejando solicitudes esperadas con tendencia a responder errores y realizar las correcciones correspondientes.

Gráfica de reeb: Para generar los gráficos de Reeb se segmentaron las peticiones de “intranet.upa.edu.mx” respecto a la cantidad de bytes enviados y la hora en que se realizó la conexión (normalizado a 24hrs) en general se puede apreciar que la mayoría de las conexiones intercambian pocos bytes.



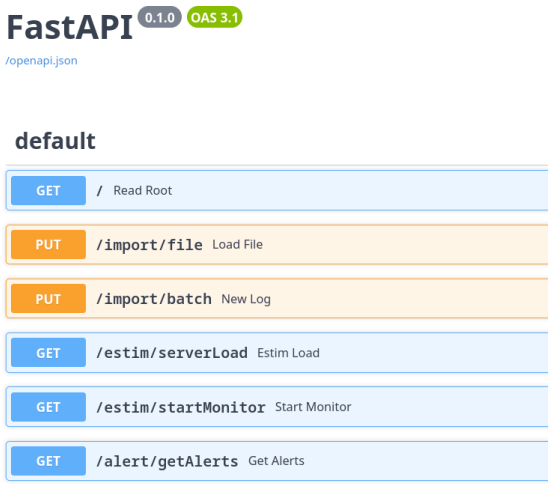
En cambio cuando generamos la gráfica de Reeb, se configuraron clusters cuya distancia máxima entre puntos es relativamente pequeña, se genera una grán cantidad de nodos (clusters), a pesar de ello logramos apreciar las estructuras correspondientes a las los rangos de 10:30 a 11:30 y 14:00 a 16:00 del gráfico de arriba en las estructuras no secuenciales del gráfico de abajo.

4. Caso de uso

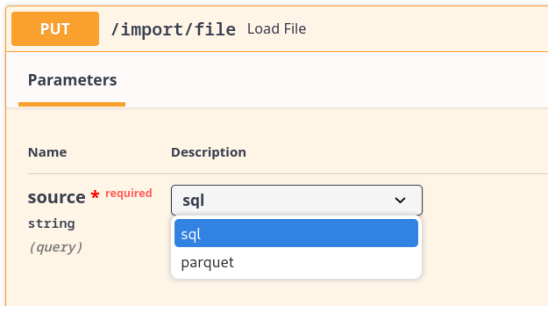
Para el desarrollo de este proyecto se realizó una API REST la cual está conectada a una base de datos que aloja la información de las conexiones que recibe un servidor.

A través de los distintos endpoints que ofrece la api es posible realizar diferentes acciones tanto de carga y reseteo de datos, realizar estimaciones y predicciones respecto a la cantidad de peticiones que se

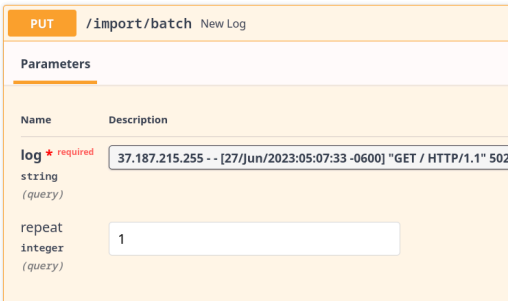
esperan dados los datos históricos, también es posible realizar monitoreo y recopilación de alertas cuando la cantidad de conexiones en un tiempo definido supera las predicciones.



Existen un endpoint para cargar la base de datos por primera vez, en su defecto, resetearla a un estado original, se puede elegir entre cargar empleando un archivo SQL o parquet.



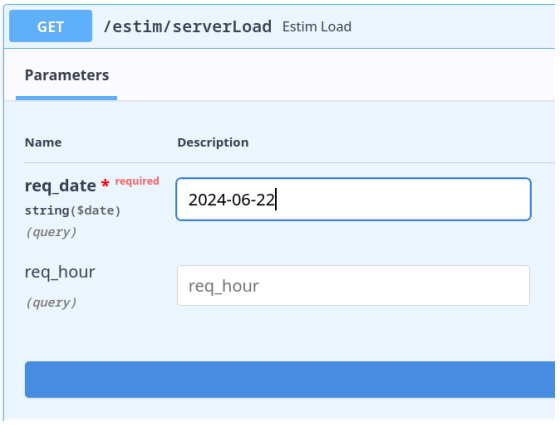
Mediante el endpoint “import/batch” es posible ingresar nuevos datos en lote, esto es útil para simular el flujo de conexiones en tiempo real, al permitir ingresar datos nuevos en la base de datos, además de que genera los datos en el estado original (crudo), siendo estos una cadena de caracteres continuos, no estructurados, que el sistema debe preprocesar para obtener información útil de la conexión. Un ejemplo de un dato original no preprocesado es *“177.249.162.144 - - [27/Jun/2023:06:01:21 -0600] "GET /image/icono/logo_upp.png HTTP/1.1" 200 67272 "https://sii.upa.edu.mx/index.php" "Mozilla/5.0 (iPhone; CPU iPhone OS 16_5_1 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/16.5.1 Mobile/15E148 Safari/604.1””* dentro de este texto se integran los datos descritos anteriormente como IP del cliente, hora, recurso solicitado, cantidad de bytes contestados, etc.



El sistema permite elegir cuántas veces queremos ingresar el mismo valor (tamaño del lote), con ello podemos simular “n” clientes solicitando el mismo recurso o el mismo dominio en un periodo de tiempo corto (hora de mayor carga del servidor).

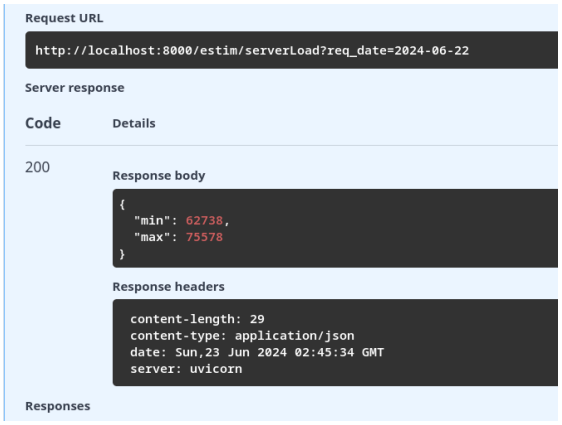
Dados los datos históricos que podemos cargar mediante los archivos sql y parquet, se pueden generar estimaciones sobre el nivel de carga (cantidad de conexiones) que se esperan al servidor dadas ciertas condiciones configurables.

Si elegimos únicamente una fecha, el sistema estimará la cantidad de peticiones que el servidor recibirá durante todo el día, por ejemplo dada la fecha “2024-06-22”

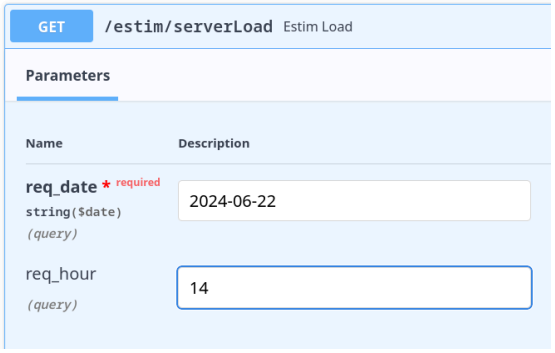


El sistema es capaz de predecir que la cantidad de conexiones será de entre 62,738 y 75,578, esto a pesar de que los datos históricos solo abarcan del 13 al 27 de junio del 2023, esto se logra dado que el sistema agrupa según el día de semana en lugar de

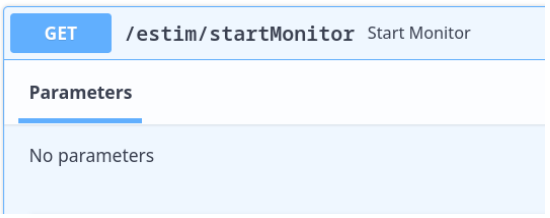
realizar regresiones lineales que en predicciones lejanas (años) pueden ser inexactas.



De igual manera es posible definir fecha y hora, en este caso se contesta la preguntá cuál será la cantidad de conexiones para el sábado 22 de junio entre las 14 y 15 horas (entre 2,500 y 3,881)

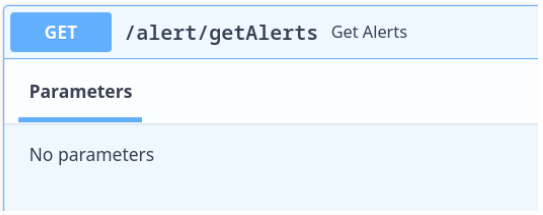


Finalmente, estas predicciones en conjunto con ingresar datos en lote, permite simular el escenario en el cual la cantidad de conexiones sea mayor a la estimada, para manejar este caso haremos uso de dos endpoints.



Que crea una tarea en segundo plano que constantemente está comparando la cantidad de conexiones recibidas con la cantidad de conexiones esperadas.

El segundo endpoint involucrado es



El cuál revisa la tabla de alertas e indica las últimas 5 existentes, así como cuanta era la cantidad esperada, la cantidad real y la hora y fecha en que se detectó la diferencia.

Cabe destacar que este proyecto se basa en tecnología de contenedores, por lo que es posible replicar el sistema en múltiples ambientes.

5. Conclusiones

El desarrollo de la API REST se considera una buena herramienta con bastante potencial por explorar, actualmente la herramienta funciona con muy pocos datos (1 millón de registros) por lo que las consultas a la base de datos a la cual se le crearon índices en las columnas fecha, hora y día de semana ofrecen un performance que ronda los 100 milisegundos por consulta, pero es necesario realizar pruebas de estrés especialmente considerando que ese millón de registros únicamente abarca 15 días, por

lo que está lejos de ser una muestra representativa.

Respecto a las limitaciones del sistema presentado, por ahora se tiene una versión muy sencilla de análisis en tiempo real, pero se considera un aspecto esencial al cuál se le pueden aplicar soluciones más robustas así como en el sistema de alertas, el cuál actualmente solo funciona si el cliente solicita saber las alertas, en lugar de que sea el sistema el que envíe la alerta en cuanto la detecte.

Otro aspecto importante para trabajo futuro es explorar campos como “user_agent”, “clean_path”, y “clean_query_list” los cuáles concentran información valiosa respecto a las intenciones del usuario así como describir de qué manera interactúa con los recursos del servidor, dichos campos son esenciales para detectar actividad maliciosa en lugar de actividad anómala como lo hace el presente análisis matemático y sistema desarrollado.

6. Referencias

1. *Abid, A., Manzoor, M. F., Farooq, M. S., Farooq, U., & Hussain, M. (2020). Challenges and Issues of Resource Allocation Techniques in Cloud Computing. KSII Transactions on Internet & Information Systems, 14(7).* https://itiis.org/journals/tiis/digital-library/manuscript/file/23716/TIIS%20Vol%2014,%20No%207-5.pdf
2. **Chung Yung, Chia-Ching Chen, Yu-Lan Yuan, Ching Li "A Systematic Model of Big Data Analytics for Clustering Browsing Records into Sessions Based on Web Log Data" in Journal of Computers doi: 10.17706/jcp.14.2.125-133* http://www.jcomputers.us/vol14/jcp1402-06.pdf
3. *Naidu, K. B., Prasad, B. R., Hassen, S. M., Kaur, C., Al Ansari, M. S., Vinod, R., ... & Bala, B. K. (2022). Analysis of Hadoop log file in an environment for dynamic detection of threats using machine learning. Measurement: Sensors, 24, 100545.* https://www.sciencedirect.com/science/article/pii/S2665917422001799
4. *Martin Degeling, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, & Thorsten Holz “We Value Your Privacy ... Now Take Some Cookies: Measuring the GDPR’s Impact on Web Privacy” doi: https://doi.org/10.48550/arXiv.1808.05096 https://arxiv.org/pdf/1808.05096*
5. *Muhammad Ubaid Ullah, Arfa Hassan, Muhammad Asif, Muhammad Sajid Farooq, Muhammad Saleem “Intelligent Intrusion Detection System for Apache Web Server Empowered with Machine Learning Approaches”* https://ijcis.com/index.php/IJCIS/article/view/13/3