Mitigación de Ataques DDoS en Base de Datos Mediante un Balanceador de Carga

Tupac Amaru Cartuche, Henry Lopez G., Oscar Paredes C.
Departamento de Ciencias de la Computación
Universidad de las Fuerzas Armadas, Sangolquí, Ecuador
mtcartuche@espe.edu.ec, hflopez@espe.edu.ec, ogparedes@espe.edu.ec

Resumen—**La incidencia de los ataques de denegación de servicio (DDoS) es alta actualmente en Internet. Varios tipos de ataques comprometen el correcto funcionamiento de los distintos componentes de los sistemas. Uno de estos son las bases de datos, importantes repositorios de información empresarial de alta importancia debido a su naturaleza en cuanto al almacenamiento de datos. Afectar el funcionamiento de un gestor de base de datos puede indisponer en gran forma el funcionamiento de una organización. El presente trabajo examina de forma experimental el uso de un middleware para mitigar este tipo de ataques. Se construye una infraestructura de pruebas en la cual se simula altas tasas de llamadas a un aplicativo Web con el objeto de colapsar una base de datos Postgresql. El alto número de llamadas hace que el gestor de base de datos colapse el servidor. A continuación se realizan pruebas con el middleware PGBouncer que funciona como balanceador de carga y se mide la efectividad de interponerlo entre el aplicativo y el motor de base de datos. Se realizan pruebas con la ayuda de Jmeter simulando varios usuarios concurrentes y validando los efectos en el porcentaje de uso de recursos en el servidor donde se encuentra instalado Postgresql.**

Palabras Clave— Postgresqsl; balanceo de carga; PGBouncer; Ataque de denegación de servicios.

I. INTRODUCCIÓN

Hoy en día empresas y personas dependen en gran medida de la tecnología para realizar actividades que van desde poner en marcha un plan de negocios hasta sencillas actividades para el hogar. Esta dependencia se debe a la facilidad de acceso a la información y a la necesidad cada vez más creciente de contar con servicios tecnológicos que faciliten la ejecución de tareas. Es por esta razón que la tecnología, conformada por hardware y software, debe ser diseñada para soportar acceso concurrente y a gran velocidad hacia los repositorios de información, sin que esta modalidad de uso represente un problema de rendimiento más aún si esta tecnología es utilizada para brindar un servicio a nivel mundial que puede ser consumida por pequeñas y grandes empresas.

Es importante mencionar algunos estudios para entender de mejor manera cómo evoluciona la tecnología. Según estudios realizados por el Instituto Nacional de Estadísticas y Censos del Ecuador (INEC), en el 2012 un censo determinó que el 26.4% de hogares posee computadores y el 13.9% tienen computadores portátiles. Para el año 2013 se destaca que las personas que tienen computador portátil corresponden al 4.2% y solo un 0.9% para aquellas que poseen un computador de escritorio [1].

Por otro lado, si se analiza el uso de Internet se observa que las personas entre 16 y 24 años representan el 64.9% seguido de las personas entre 25 y 34 con el 46.2%, lo que evidencia claramente quienes son los usuarios de Internet según los datos de las últimas estadísticas del INEC para el año 2012. Para el año 2013 el crecimiento es de 3.70% y se pronostica que para el año 2014 será del 5.30%. Como dato adicional importante se debe destacar que el 50.9% de la población usa Internet en el hogar y el 26.6% accede en centros públicos [2].

Conforme avanza la innovación en tecnología, también se observa el uso cada vez más frecuente de teléfonos celulares inteligentes para acceder a diferentes servicios e información desde cualquier parte de mundo. Es así que el 4.70% adquiere líneas celulares, mientras que líneas fijas decrece en un 2.80% según el INEC 2013[1].

Del análisis presentado se observa que cada vez es más fácil acceder a Internet y que las empresas y personas son vulnerables en este espacio, su información privada está expuesta si no se toman las precauciones debidas. Las bases de datos que son los mecanismos implementados para resguardar la información de empresas, clientes y personas, son blanco de ataques especializados con el objetivo de obtener o manipular información. Muchas veces el objetivo del atacante es dejar sin servicio una página o sistema web colapsando su base de datos mediante DDoS.

El presente estudio pretende demostrar cómo las herramientas que controlan o balancean las conexiones a la base de datos ayudan a mitigar uno de los ataques más comunes conocidos, el ataque DDoS. Para las pruebas se utilizó una base de datos PostgreSQL y el balanceador de carga PGBouncer para administrar las conexiones a la base de datos. Adicionalmente se utilizó el software JMeter para ejecutar pruebas de estrés y verificar peticiones al servidor, diagnosticando el comportamiento de conexiones entrantes y aplicando acciones oportunas y proactivas según amerite.

En la sección II se presentan conceptos relacionados con base de datos PostgreSQL, PGBouncer, JMeter y ataques DDoS. En la sección III se presenta el diseño y la implementación del experimento, pasando a analizar y evaluar los resultados obtenidos en la sección IV. En la sección V se mencionan trabajos relacionados y para culminar se presentan la conclusiones y trabajos futuros en la sección VI.

II. MARCO TEÓRICO

A. *Base de datos PostgreSQL*

PostgreSQL es un ORDBMS (Object Relational Database Management System), un sistema de gestión de base de datos relacionales que se encuentra disponible en el mercado desde hace tres décadas y que es distribuido bajo licencia BSD, es decir, una licencia de software libre.

Este sistema de gestión se ha desarrollado a la par con otros gestores de tipo comercial y propietario, de manera que en la actualidad tanto sistemas propietarios como libres comparten gran popularidad a nivel mundial, adjudicándose el 55% las soluciones propietarias y el 45% aquellas que son de libre distribución, según el estudio presentado por la empresa DB-Engines en [3].

Sin embargo, a pesar de que las soluciones propietarias mantienen una hegemonía ligeramente superior sobre las soluciones libres, de mantenerse la tendencia mundial de los últimos años se podría asegurar que en los próximos años las soluciones libres para gestionar bases de datos serán más utilizadas por muchas empresas a nivel mundial (Figura 1).

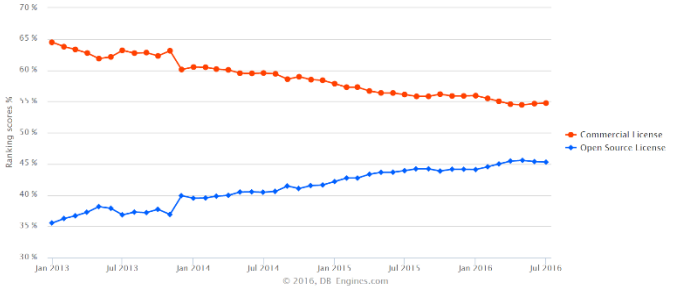

Figura 1: Tendencia de popularidad a Julio 2016. Publicado en [3].

Considerando el panorama presentado en los párrafos anteriores, no es descabellado pensar que los ciber delincuentes orienten sus ataques hacia aquellas empresas que utilicen gestores libres, más aún cuando son ampliamente utilizadas tal como es el caso de PostgreSQL que ocupa el primer lugar en soluciones libres para base de datos relacionales como lo demuestra DB-Engines.

Con esta perspectiva, es necesario buscar mecanismos de protección contra ataques que impidan el normal funcionamiento de los servidores de base de datos, herramientas que ayuden a mejorar y garantizar la seguridad de la información.

B. *Balanceador de carga: PGBouncer*

Un balanceador de carga es un mecanismo de hardware y software utilizado para gestionar una gran cantidad de solicitudes de usuario y dividir de manera equitativa el trabajo entre los recursos disponibles de manera que se eviten cuellos de botella [5].

Para el caso de PostgreSQL, existe un software que administra la cantidad de conexiones entrantes hacia el servidor de base de datos, consiguiendo disminuir y controlar las solicitudes de acceso, reduciendo de esta manera el tiempo de procesamiento y el tiempo de uso de los recursos del servidor [4], gestionando conexiones hacia una o más bases de datos.

PGBouncer soporta tres tipos de pool de conexiones: 1) Pool de sesiones, 2) Pool de transacciones, y; 3) Pool de Sentencias. En el primer caso, se asigna un conexión para cada cliente para todo el tiempo que dure la conexión; en el segundo caso se asigna una conexión por cada transacción que se ejecute, y; en el último caso se restringe la conexión a nivel de sentencias.

Dadas las tres posibilidades o modos de funcionamiento descritos en el párrafo anterior, es posible utilizar el balanceador instalándolo entre el servidor de aplicaciones y el servidor de base de datos para controlar el número de conexiones, autorizando o negando aquellas que puedan ser generadas como parte de un ataque DDoS.

C. *Ataque de denegación de servicio (DDos)*

Un ataque de denegación de servicio distribuido (DDoS) es un procedimiento que atenta contra los recursos de red de un servidor principalmente web y de base de datos, dejando no disponible el servicio para los usuarios de forma temporal o indefinidamente.

Los ataques de denegación de servicios se han desarrollado y ejecutado sobre todas las capas del modelo OSI, empezando por ataques de red hasta ataques a sesiones y en la capa de aplicación hoy en día. Este escalamiento en las diferentes capas del modelo OSI hace que se incremente la capacidad de detectar ataques [5]. Si a lo anterior se complementa con aplicaciones mal diseñadas, que no cumplen buenas prácticas internacionales como OWASP, representan una gran vulnerabilidad que puede ser utilizada por los ciber atacantes.

D. *Generador de carga JMeter*

Software de generación de carga que permite evaluar el desempeño de un servicio o sistema cuando es expuesto a condiciones de gran demanda, principalmente aquellas disponibles en sistemas web.

Inicialmente JMeter fue diseñado solamente para ejecutar pruebas de estrés en aplicaciones web, sin embargo en la actualidad es capaz de realizar desde una petición sencilla hasta una secuencia de peticiones al servidor que permiten diagnosticar el comportamiento de una aplicación en condiciones de producción [6].

III. DISEÑO E IMPLEMENTACIÓN DE LA SOLUCIÓN

A. *Diseño e implementación de la topología de prueba*

Tomando en cuenta que varios de los tipos de ataques DDoS tienen como efecto el consumo desmesurado de recursos de un sistema, se diseñó una topología basada en aplicaciones web para realizar la experimentación.

Se utilizó la arquitectura clásica de las aplicaciones basadas en Internet, configurando un servidor web en el cual se ejecutó un aplicativo constituido de una interface de usuario con varios controles. A continuación se interpuso una capa de lógica de negocio, encargada de transportar los datos desde la interface de usuario hacia la capa de acceso a datos, la que se encargó de manejar la conexión con la base de datos.

En otro servidor se instaló un gestor de base de datos. El motor de base de datos elegido para la experimentación fue

Postgresql, software Open Source que una alternativa de gestor de base de datos altamente recomendable para pequeñas y medianas empresas u organizaciones. Se destaca por su alta capacidad de trabajo y su facilidad de mantenimiento.

Los dos servidores fueron desplegados en un entorno virtualizado ejecutándose en un computador. Los elementos del entorno de prueba se resumen en la siguiente tabla (Tabla I):

TABLA I. EQUIPO USADO EN EL EXPERIMENTO

Cantidad	**Componente**	
	Nombre	***Características***
1	PC 01	Computador Host Intel CORE i5, procesador 2.5Ghz, 2 núcleos, 4 procesadores lógicos, 8 Gb RAM, Windows 10 64 bits.
2	VPC 01	Servidor virtual Intel CORE i5, procesador 2.5Ghz, 1 procesador, 1.5 Gb RAM, Ubuntu 15.10 32 bits
3	VPC 01	Servidor virtual Intel CORE i5, procesador 2.5Ghz, 1 procesador, 1.5 Gb RAM, Fedora 19 (Schrödinger's Cat) 64 bits
4	WiFi	D-Link

Las herramientas de software que fueron utilizadas para el experimento permitieron realizar las tareas de simulación de ataque al aplicativo web desplegado en la infraestructura, medir los efectos de los ataques y mitigar el ataque en base de datos. Se cuenta así con los siguientes componentes (Tabla II):

TABLA II. HERRAMIENTAS DE SOFTWARE USADAS EN EL EXPERIMENTO

Herramienta	
Nº	***Detalles***
1	VMware® Workstation 12 Pro V. 12.0.0 build-2985596
2	Servidor Web Apache 2.4
3	Postgresql 9.3
4	PGBouncer 1.7.2
5	Apache JMeter 2.13
6	Sysstat 11.2.1.1 released (stable version).
7	PHP 5.0
8	Eclipse IDE for PHP Developers Mars .1 Release 4.5.1

Se muestra a continuación la topología de la aplicación web que se ha empleado en el experimento en sus dos fases: 1) sin PGBouncer y, 2) con PGBouncer.

El primer experimento (Figura 2) muestra la configuración de una topología clásica de aplicaciones web en una máquina virtual Ubuntu con un servidor de aplicaciones Apache desarrollada en PHP 5. Este experimento permitió obtener la línea base de medición de resultados al experimentar llamadas hacia la base de datos residente en el servidor Fedora.

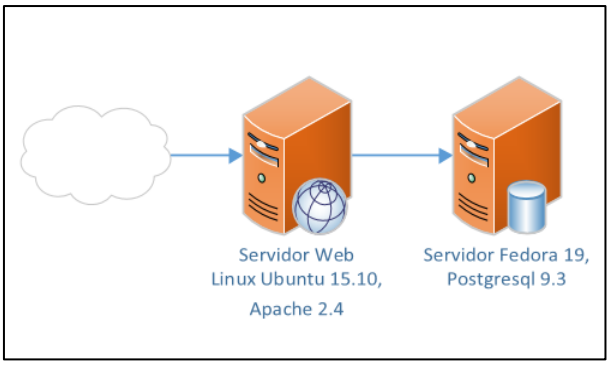

Figura 2. Topología sin PGBouncer

En el segundo experimento, en el cual se planteó la mitigación al problema de ataques DDoS, se mantuvo la misma configuración del primer experimento, agregando a la arquitectura el aplicativo PGBouncer el cual fue desplegado entre el servidor web de aplicación y el servidor de base de datos.

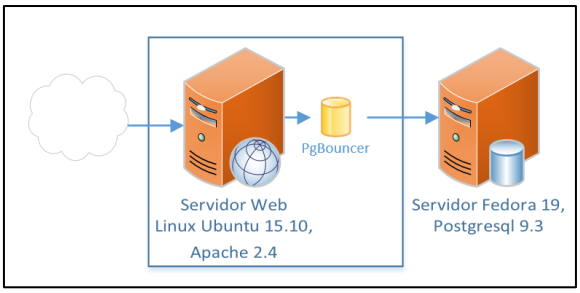

Figura 3. Topología con PGBouncer

PGBouncer funcionó a modo de gestor de conexiones, siendo un middleware entre la aplicación y el servidor de base de datos, administrando el pool de acceso como se aprecia en la Figura 3.

B. Configuración del ataque

Para la realización del ataque, se construyó el aplicativo web dvdrental, con dos páginas PHP. En la primera (/dvdrental/busqueda.php) se incluyó el ingreso de datos para la consulta. Una captura de pantalla se muestra en la Figura 4.

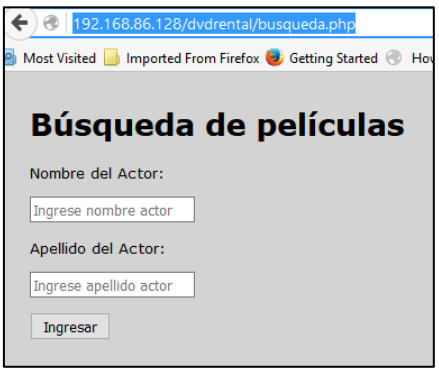

Figura 4. Página busqueda.php

Por medio del botón "ingresar" las variables de búsqueda se dirigen a la segunda página php que realiza la consulta a la base de datos (/dvdrental/busquedaAction.php?) En esta se ejecuta la

consulta mostrada en la Figura 5:

```
select distinct f.title,
                f.description,
                a.first_name ||' - ' || a.last_name as actor
        from    film f, film_actor fa, actor a
        where   f.film_id = fa.film_id
            and fa.actor_id = a.actor_id
            and a.first_name ilike'%a%'
            and a.first_name ilike '%a%'
```

Figura 5. Consulta SQL de prueba

Las tablas utilizadas para construir la consulta de prueba pertenecen a la base de datos "dvdrental" creada en el servidor. Esta es una base de datos Postgresql de prueba descargable con fines académicos [7].

El modelo de consulta incluyó combinaciones de tres tablas (film, film_actor y actor) agregando un filtro por medio del comando "ilike". La ejecución de forma concurrente de esta consulta indispuso los servicios en la infraestructura, provocando demoras y fallas en la ejecución de los comandos en la base de datos y por tanto, fallas en el sistema en general.

La conexión a la base de datos desde el aplicativo desplegado en el servidor Apache se la realizó de forma directa a través de la siguiente cadena de conexión: (Figura 6)

```
host='192.168.86.130' port='5432' dbname='dvdrental'
user='usuario' password='password'
```

Figura 6. Cadena de conexión a base de datos directa

Para realizar el ataque de denegación de servicio, se utilizó la herramienta Apache JMeter, a través de la cual se configuró 5 escenarios de prueba para validar el experimento. Cada escenario estaba representado por 500, 1000, 1500, 2000 y 2500 hilos respectivamente. Cada hilo representó un usuario virtual realizando cada uno de ellos 1 transacción sobre la aplicación web.

C. *Propuesta de mitigación*

En el caso del ataque de denegación de servicio ejecutado en el experimento y su afectación al servidor de base de datos, se propuso el uso del middleware PGBouncer. El mismo se instaló en el servidor de aplicaciones Ubuntu como se describe en la Figura 7.

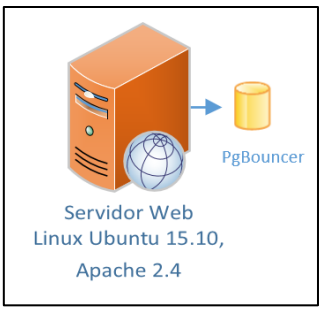

Figura 7. PGBouncer en el servidor de aplicaciones

Los parámetros de configuración se encuentran en el archivo */etc/PGBouncer /PGBouncer .ini* en el cual se definió parámetros como la cadena de conexión a la base de datos remota, la dirección IP y el puerto de escucha para conexiones locales, el tipo de autenticación, el archivo de usuarios, el modo de Pool de conexiones, el máximo número de conexiones desde el aplicativo hacia PGBouncer y desde PGBouncer hacia la base de datos remota, entre otros parámetros (Figura 8).

```
[databases]
dvdrental= host=192.168.86.130 port=5432 dbname=dvdrental
user=[redacted] pool_size=80

[pgbouncer]

logfile = /var/log/postgresql/pgbouncer.log
pidfile = /var/run/postgresql/pgbouncer.pid
listen_addr = 127.0.0.1
listen_port = 6432
auth_type = trust
auth_file = /etc/pgbouncer/userlist.txt
pool_mode = transaction
server_reset_query = DISCARD ALL
max_client_conn = 300
default_pool_size = 20
server_lifetime = 1200
server_idle_timeout = 60
client_idle_timeout = 60
```

Figura 8. Configuraciones PGBouncer

De esta forma, instalado el software propuesto de mitigación para el ataque experimental, se configuró la cadena de conexión del aplicativo de la siguiente manera (Figura 9):

```
host='127.0.0.1' port='6432' dbname='dvdrental'
user='usuario'
```

Figura 9. Configuraciones PGBouncer

Por medio de esta cadena de conexión, la aplicación web de prueba se conectó a la base de datos a través de PGBouncer y no directamente al servidor. Se ejecutó el mismo escenario de pruebas descrito en la Configuración del Ataque.

En la ejecución, PGBouncer creó un pool de conexiones hacia la base de datos remota generando un balanceo de carga en la ejecución de consultas hacia la base de datos; esto ayudó a mitigar la saturación de recursos en el servidor de base de datos Postgresql.

D. *Pruebas y medición*

Realizadas las pruebas de conformidad a lo descrito en la configuración del ataque (B) y la propuesta de mitigación (C), los resultados de las mediciones se colectaron por medio del archivo de salida de Apache JMeter en donde se obtuvo el tiempo de respuesta de las ejecuciones, la latencia, el número de bytes recibidos como respuesta, etc.

Las mediciones se guardaron en archivos .csv para cada una de las ejecuciones y se los procesó para obtener las estadísticas de tiempo de respuesta promedio para cada uno de los experimentos sin el uso de PGBouncer y usando PGBouncer para mitigar el ataque.

Para verificar los parámetros de funcionamiento de las pruebas iniciales en el servidor de aplicaciones y base de datos se utilizó el aplicativo sysstat tanto para medición de uso de CPU como para medir el porcentaje de consumo de memoria. Se usó comandos similares a los de la figura 10, recopilando los resultados para su posterior análisis:

```
sudo sar -r 1 5000 > cpb_100_100_100_mem.txt (Memoria)
sudo sar -u 1 5000 > cpb_100_100_100_cpu.txt (CPU)
```

Figura 10. Configuraciones PGBouncer

IV. Evaluación de resultados y discusión

Como producto de la experimentación se obtuvieron mediciones en parámetros como tiempo promedio de respuesta, porcentaje de éxito de las búsquedas, porcentaje de uso de CPU y de memoria en el servidor de base de datos.

Para tiempo promedio de respuesta (Tabla III), los tiempos de ejecución en el servidor de aplicaciones aumentaron cuando la solución trabajó con PGBouncer con relación a cuándo no se utilizó el mencionado middleware.

TABLA III Tiempos Promedios de Respuesta

	Tiempo Promedio Respuesta (ms)	
Usuarios Concurrentes	**SIN PG BOUNCER (Usarios)**	**CON PG BOUNCER (Usuarios)**
500	1033,32	6331,31
1000	8895,00	9291,74
1500	9227,24	10004,15
2000	28859,21	40259,05
2500	37140,60	47678,43

Esto se debe a que cuando trabajó en su función de balanceador, el programa PGBouncer generó un pool de conexiones por medio del cual se canalizaron las instrucciones SQL (Figura 11).

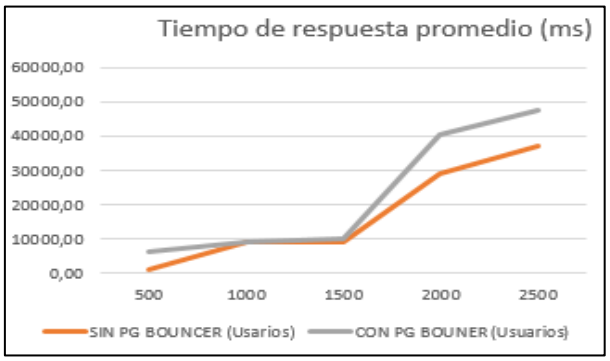

Figura 11 Tiempo de respuesta promedio

Cuando masivamente se enviaron peticiones al servidor de base de datos, PGBouncer detuvo el tráfico a la base de datos y realizó las consultas conforme a la configuración de máximas conexiones configurado en el archivo pgbuncer.ini. Esto repercutió en la respuesta de las páginas.

En cuanto al éxito de las ejecuciones se verificó y registró en la Tabla IV que el porcentaje de éxito es mayor cuando se usó el balanceador de carga, esto debido a que al canalizarse las ejecuciones por el pool de conexiones, este detuvo las transacciones y las realizó ordenadamente aplicando el balanceo de carga.

TABLA IV Tiempos Promedios de Respuesta

	Porcentaje de Éxito (sobre 100%)	
Usuarios Concurrentes	**SIN PG BOUNCER (usuarios)**	**CON PG BOUNCER (Usuarios)**
500	100,00	100,00
1000	100,00	100,00
1500	88,01	88,96
2000	55,68	72,59
2500	50,12	53,24

En la Figura 12 también se aprecia que las ejecuciones exitosas desde el servidor de aplicaciones se vieron comprometidas en este caso para un ataque de denegaciòn de servicios a pesar de que hubo una mejora con el uso de PGBouncer .

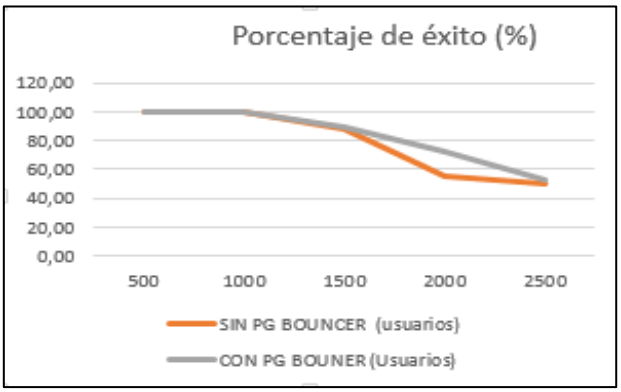

Figura 12.Porcentaje de éxito

En lo referente a los porcentajes de uso de procesador, se obtuvo datos que reflejan que no existió una gran diferencia entre las medidas con el balanceador de carga y sin el mismo en el servidor de base de datos que es de donde se obtuvieron estas medidas (Tabla V).

TABLA V Porcentaje de uso de CPU

	% Uso de CPU	
Usuarios Concurrentes	**SIN PG BOUNCER (usuarios)**	**CON PG BOUNCER (Usuarios)**
500,00	16,30	16,20
1000,00	16,85	16,52
1500,00	17,52	18,20
2000,00	21,30	17,21
2500,00	22,50	20,90

El comportamiento del porcentaje de uso de CPU para el servidor de base de datos cuando estuvo sin ningún tipo de actividad fue de hasta el 2 %, por lo que los porcentajes obtenidos luego del experimento nos indicaron de todas maneras que se aumentó el uso de este recurso (Figura 13).

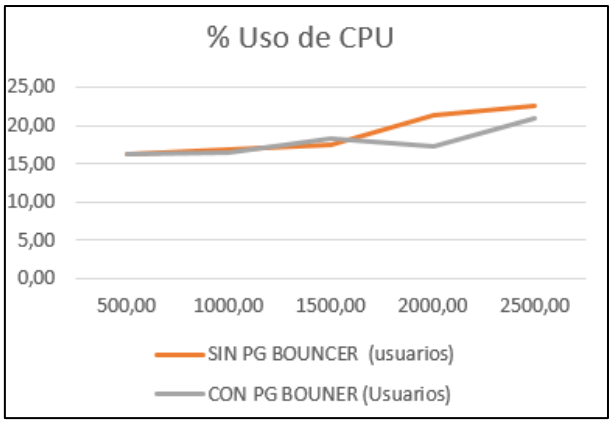

Figura 13.Porcentaje uso de CPU

Al realizarse la experimentación, sin PGBouncer se alcanza porcentajes de uso superiores al 95% de memoria RAM ocasionando mal funcionamiento de la base de datos, indisponiendo el servicio (falta de conexión a la base de datos) o en el mejor de los casos haciendo las consultas mucho más lentas.

Usando PGBouncer se aprecia una disminución en el porcentaje de uso de memoria en el servidor de base de datos, conforme a las mediciones realizadas. (Tabla IV).

TABLA VI PORCENTAJE DE USO DE CPU

	% Uso de Memoria	
Usuarios Concurrentes	**SIN PG BOUNCER (Usarios)**	**CON PG BOUNCER (Usuarios)**
500,00	93,51	87,90
1000,00	94,51	87,80
1500,00	95,21	88,90
2000,00	95,20	84,57
2500,00	95,23	91,10

El funcionamiento de la memoria en el servidor de base de datos registrado durante el experimento se puede apreciar en la Figura 14.

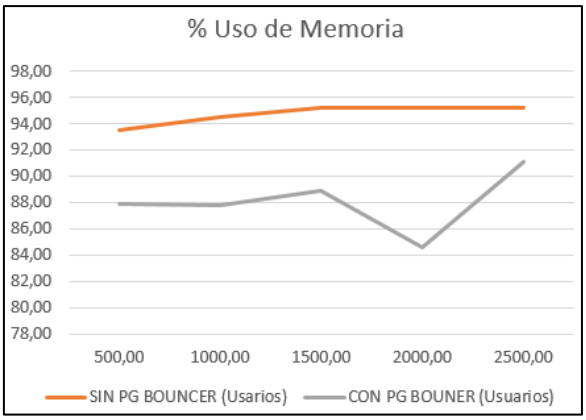

Figura 14.Porcentaje uso de CPU

V. TRABAJOS RELACIONADOS

Ataques DDoS son comunes en el entorno de Internet siendo una de las razones el bajo precio que se puede pagar por realizar un ataque. Por tal motivo constantemente se emplea esfuerzo en la búsqueda de mecanismos de defensa y mitigación de este tipo de ataques, mecanismos necesarios a nivel de aplicación, de red de datos y de base de datos.

Considerando que una base de datos de cualquier organización almacena información valiosa para su propietario, el presente trabajo constituye un aporte hacia la implementación de mecanismos de defensa a nivel de base de datos que sean de bajo costo y que utilicen PostgreSQL como su sistema gestor de base de datos, tomando en cuenta como se describió en la sección II de este artículo, que PostgreSQL se está convirtiendo en una solución libre ampliamente aceptada a nivel mundial.

Existen estudios importantes sobre mitigación de ataques DDoS, Santanna, J. J., Durban, R., Sperotto, A., & Pras, A, exponen en [8] un análisis e identificación de las características de algunos sitios web denominados Booters con el objetivo de facilitar trabajos futuros, en vista que estos Booters están facilitando la ejecución de ataques DDoS en Internet. En [9] se propone un mecanismo de mitigación basado en redundancia de tablas, de manera que el servicio siempre esté activo. Con igual objetivo, Yujie, Z. H. A. O., Bhogavilli, S., & Guimaraes, R, proponen en [10] un sistema implementado por computador para detectar en base a mensajes transmitidos de solicitud y respuesta las direcciones IPs de los atacantes y ejecutar procesos de bloqueo a los atacantes. Y en lo que respecta a mitigación de ataques en redes, en [11] se propone un algoritmo para detectar y mitigar ataques en redes Wireless 3G/4G.

La mayoría de trabajos se han enfocado en soluciones a nivel de red, de servidores web y aplicaciones, por lo que el presente artículo complementa los estudios realizados y aporta a conseguir mayor protección contra ataques de denegación de servicio.

VI. CONCLUSIONES Y TRABAJOS FUTUROS

Las medidas de mitigación frente a ataques DDoS que deben tomar los administradores de sistemas, deben apuntar a todos los componentes de una aplicación. Uno de los componentes críticos en una arquitectura de sistemas es la base de datos, por lo que su correcto funcionamiento y seguridad se vuelve un aspecto crítico. Muchas veces no se contempla implementar soluciones en base de datos, enfocándose más en la seguridad de los servidores Web, en los firewalls y otros tipos de medidas de protección.

Por medio de la experimentación expuesta, se ha demostrado que mediante el uso de administradores de conexiones como PGBouncer, además de optimizar el funcionamiento de las aplicaciones, ayuda a mitigar ataques DDoS contra la base de datos. Los resultados de consumo de recursos en el servidor indicaron una mejora, evitando el colapso de la aplicación en conjunto.

El desempeño de la herramienta de mitigación examinada en el experimento tuvo como contraparte en el servidor de aplicaciones, un aumento del tiempo de respuesta en el

despliegue de las consultas a la base de datos. Esto nos indica que a pesar de que se mitigó el efecto del ataque en el servidor de base de datos, debe de tomarse medidas complementarias en los otros componentes de la arquitectura de un sistema, para que toda la aplicación trabaje en forma adecuada.

Como trabajo futuro se pretende atacar esta última problemática examinando la posibilidad de implementar a parte de la utilidad para realizar balanceo de carga una solución basada en un WAF u otras implementaciones en Firewall, así como mejoras en los métodos de prevención de ataques de SQL Injection entre otras optimizaciones.

REFERENCES

[1] INEC (2013) Disponible en: http://www.ecuadorencifras.gob.ec/tecnologias-de-la-informacion-y-comunicacion-tic/

[2] Disponible en: http://es.slideshare.net/agenciavertice/anlisis-de-estadsticas-internet-y-redes-sociales-de-ecuador-a-junio2014-por-elerick

[3] DB-Engines, "Popularity of open source DBMS versus commercial DBMS" [online], 2016. Disponible en: http://db-engines.com/en/ranking_osvsc.

[4] Squicciarini, A. C., Paloscia, I., & Bertino, E. (2008, April). Protecting databases from query flood attacks. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on* (pp. 1358-1360). IEEE.

[5] EnterpriseDB Corporation, How to Set Up PgBouncer Connection Pooling for Postgres Plus Standard Server, 2010, Disponible en: http://get.enterprisedb.com/docs/Tutorial_All_PPSS_pgBouncer.pdf

[6] Holmes, D. (2013). Mitigating DDoS Attacks with F5 Technology. *F5 Networks, Inc*.

[7] Postgresql Tutorial. Disponible en: http://www.postgresqltutorial.com /postgresql-sample-database/

[8] Mahajan, D., & Sachdeva, M. (2013). DDoS Attack Prevention and Mitigation Techniques-A Review. *International Journal of Computer Applications*, *67*(19).

[9] Santanna, J. J., Durban, R., Sperotto, A., & Pras, A. (2015, May). Inside booters: an analysis on operational databases. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on* (pp. 432-440). IEEE.

[10] Silva, Mauro., Romero, Diego.,Bastidas, Christian., & Fuertes, Walter. (2015, Noviembre). Mitigacion de Ataques DDoS a traves de Redundancia de Tablas en Base de Datos. In Memorias VIII Congreso Iberoamericano de Seguridad Informatica, 2015, on (pp.56-62).

[11] Yujie, Z. H. A. O., Bhogavilli, S., & Guimaraes, R. (2014). *U.S. Patent No. 8,869,275*. Washington, DC: U.S. Patent and Trademark Office.

[12] Gupta, A., Verma, T., Bali, S., & Kaul, S. (2013, January). Detecting MS initiated signaling DDoS attacks in 3G/4G wireless networks. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*(pp. 1-60). IEEE.