Optimización de Software con IAO

Rolando Ríos Ramos
Maestría en Ingeniería en Ingeniería de Computadoras
Nelliud Torres, PhD.
Departamento de Ingeniería Eléctrica, de Computadoras y Ciencias de Computadoras
Universidad Politécnica de Puerto Rico

Abstracto *— La proliferación de dispositivos interconectados en ambientes de redes, en conjunto con la multiplicidad de avances tecnológicos, ha ocasionado que las aplicaciones y sistemas se puedan tornar obsoletos en menor tiempo. Como consecuencia, las compañías enfrentan el reto de intentar prolongar la vida útil de sus aplicaciones. Los altos costos de licenciamiento, equipos y contrataciones, entre otros factores, impiden a compañías y usuarios domésticos remplazar sus sistemas, por lo que muchas compañías optan por optimizar los mismos. Llevar a cabo un proyecto de optimización puede ser una tarea compleja. No tan solo se lidia con situaciones técnicas sino que factores como tiempo y presupuesto afectan de manera directa el alcance del proyecto. Son muchas las variables que impiden aplicar la misma solución en diferentes proyectos por lo cual una metodología adecuada es indispensable. El proceso desarrollado "Inventario, Análisis y Optimización" propone ser una herramienta adecuada para esos fines.*

Términos claves *— Análisis, AngularJS, HTML/PHP, Inventario, Optimización, Pruebas, Rendimiento, Seguridad, SQL, Utilización.*

INTRODUCCIÓN

Actualmente, muchas compañías intentan extender el tiempo de vida útil de un Software implementado proyectos de optimización en lugar de remplazar los mismos. En ocasiones se trata de programas diseñados internamente y en otras, aplicaciones que ya no tienen soporte técnico con el manufacturero. La implementación de nuevos sistemas no tan solo tiene un impacto por costos de licenciamiento y equipos, sino que también incluye altos costos directos e indirectos, entre éstos los servicios profesionales. Esto conlleva una inversión sustancial para las compañías, que no pueden asumir con frecuencia. Por estas y otras razones, algunas compañías optan por optimizar sistemas existentes para extraer el máximo de su inversión y evitar la obsolescencia de sus sistemas. La obsolescencia de Software puede poner en riesgo la preservación de datos debido a razones de vulnerabilidad a fallas en el sistema así como ataques internos o externos [1]. El incremento en el tamaño de la Base de Datos, conjuntamente con el aumento de operaciones, tiende a reducir el desempeño de los sistemas. Aquellos con deficiencias en el diseño, son los más expuestos. Aunque los sistemas se tornen obsoletos, los datos almacenados en los mismos pueden tener una vida útil mucho más extensa. Es vital para todo tipo de compañía mantener operables este tipo de sistemas, aun cuando estos hayan sido remplazados u operen de forma paralela con aplicaciones más actualizadas.

El análisis de este trabajo, responde a una típica situación que se vive en la industria, donde se solicita mejorar el desempeño de un Software implementado. Este tipo de proyectos suelen contar con menos tiempo y recursos presupuestarios

Para simplificar la complejidad de un proyecto de optimización, se propone el uso del proceso IAO. Este pretende simplificar el entendimiento de la ingeniería de procesos requerida al momento de llevar a cabo un plan de optimización de Software.

Tendencias en la Industria de Software

Las tendencias actuales en la industria revelan que los Software Empresariales siguen manteniendo un incremento en su demanda, específicamente los sistemas de planificación de recursos empresariales (ERP por sus siglas en inglés) [2]. Los sistemas ERP típicamente manejan la producción, logística, distribución, inventario, envíos, facturas y contabilidad de la compañía de forma modular. Este

tipo de Software es utilizado por diferentes compañías y agencias de gobierno. La implementación de un ERP es usualmente una inversión sumamente costosa la cual ronda en los $6.1 millones con una duración de 15.7 meses [3]. Definitivamente, una inversión de esta índole debe ser por largo término y no puede ser realizada con frecuencia.

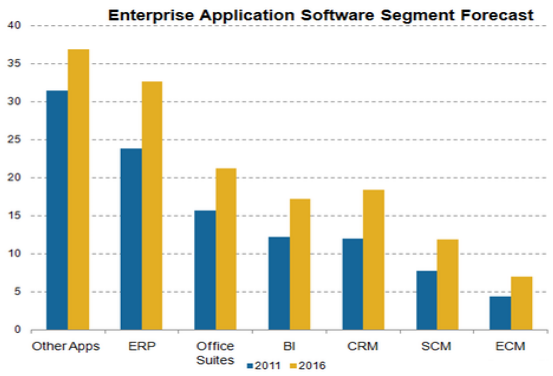

Figura 1
Ingresos de Software Empresarial Mundial en 2011 y 2016, según el Segmento (en billones)

La figura 1 muestra una comparación entre el año 2011 y los ingresos estimados por venta de software empresarial a nivel mundial en el 2016.

Teniendo en cuenta la cantidad de ingresos que genera este tipo de productos, las expectativas del negocio de soporte técnico y optimización de Software son muy prometedoras.

PROCESO IAO

IAO (Inventario, Análisis y Optimización) es un proceso analítico propuesto en este proyecto que pretende simplificar el entendimiento de la ingeniería de procesos requerida al momento de llevar a cabo un plan de acción para mejorar el desempeño de un sistema. IAO está inspirado en el proceso "Ciclo de vida de desarrollo de Software" (SDLC) por sus siglas en ingles."

El Ciclo de Vida de Desarrollo de Software en la ingeniería de sistemas es el proceso de creación o modificación de los sistemas, modelos y metodologías que se utiliza para desarrollar Software [4]. Las metodologías constituyen el marco para la planificación en el proceso de desarrollo o modificación de Software.

Existen diversos modelos y metodologías SDLC pero en general estos consisten de la siguiente serie de fases:

- **Inicialización**: Durante la fase inicial, la organización establece las necesidades del sistema y documenta el propósito.
- **Definición de Requisitos**: Se determinan los requerimientos usualmente utilizando métodos interactivos tales como muestreos, consultas a usuarios, investigación de situaciones reportadas, entre otras.
- **Análisis de Necesidades:** Una vez reunida toda la información referente a los requerimientos, se clasifica y documenta la que impacta al proyecto.
- **Diseño:** Se elaboran los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis. También se toma en consideración una arquitectura de seguridad de sistemas
- **Desarrollo del Software:** Durante esta fase se diseña, compra u optimiza el Software.
- **Prueba de sistemas:** El sistema se emplea de manera experimental para asegurar que el Software no tenga fallas, es decir que funciona de acuerdo con las especificaciones y en la forma en que los usuarios esperan que lo realice.
- **Implantación y Evaluación:** En la fase de implementación, la organización configura y habilita las funciones de seguridad del sistema, pone a prueba las funcionalidades y obtiene una autorización oficial para operar en ambiente de producción.
- **Operación y mantenimiento:** En esta fase, los sistemas ya están en su lugar y funcionando, incluyendo modificaciones y optimizaciones. Constantemente se observa el desempeño y seguridad del sistema.

En el modelo propuesto, estos pasos se simplifican en tres fases con el propósito de implementar una metodología de desarrollo rápido y simple sin perder de perspectiva un diseño sólido y

estructurado. Aunque la metodología del proceso está enfocada en la optimización de un Software, esta puede ser aplicada en la implementación o desarrollo. Las 3 fases requeridas en IAO son las siguientes:

- **Inventario**: Esta fase consiste en identificar todos los requerimientos funcionales incluyendo situaciones reportadas en conjunto de un inventario de recursos de sistema existentes. El inventario de recursos de sistema, debe ser un componente de los equipos y cualquier tipo de Software que la compañía reporte como parte de sus activos, conjuntamente con la a corroboración de los recursos encontrados. Todos los inventarios deben ser documentados y discutidos con el propósito de asegurar que hayan sido identificados. Es muy probable que al discutir el listado se identifiquen recursos de los cuales la compañía no tenía constancia.
- **Análisis:** La finalidad de la fase de Análisis es establecer el enfoque, alcance y objetivo del proyecto. El punto clave es identificar todas las situaciones registradas en la fase anterior con la intención de replicar los escenarios y hacer análisis de las deficiencias encontradas. Esto incluye procesos que comprometen los recursos del sistema, situaciones de riesgo, integridad de datos, posibles fallas y factores que puedan afectar la aplicación a medida que crezca el volumen de datos y transacciones. Después de identificados los puntos claves, se debe culminar la fase con una hipótesis que dé lugar a un plan de mejoras y diseño conceptual. Este debe incluir todos los diseños detallados de soluciones propuestas tomando en cuenta los recursos disponibles incluyendo factores de tiempo y presupuesto. Todos los resultados de pruebas, diseños y planes de trabajo deben ser documentados.
- **Optimización:** Teniendo en cuenta las deficiencias y necesidades analizadas en la fase anterior se procede a trabajar con el plan de mejoras. Este incluye experimentación de soluciones propuestas y diseñadas. Al igual que en la fase de Análisis, es importante poner a prueba las soluciones desarrolladas y documentar los resultados. Tanto positivos como negativos deben estar claramente documentados. El alcance de la fase de optimización no debe concluir con solo mejorar deficiencias en diseño, también se debe considerar el uso de nuevas tecnologías que pudieran llevar el nivel de optimización a uno más alto. Implementar esta medida puede extender la vida útil del Software optimizado. Al final de la fase de optimización se debe haber completado el proyecto, incluyendo el monitoreo del desarrollo puesto en producción. Un completo inventario, conjuntamente con un análisis profundo y un fuerte diseño estructurado, evitara medidas correctivas posteriores a la implementación.

EXPERIMENTACIÓN

Para propósitos de modelar la aplicación del proceso IAO en un proyecto de Optimización, se escogió un Software de código abierto, el cual es utilizado por pequeñas empresas y usuarios domésticos. La pretensión de este modelo, es experimentar con el Software obtenido para descubrir si al someter el mismo mediante el proceso IAO se puede mejorar su desempeño. El Software escogido para esta prueba es un sistema de facturación de nombre SimpleInvoice. El portal "The Hosting News", indica que este Software es muy utilizado por pequeñas empresas y usuarios domésticos alrededor del mundo; situándola en la posición número 1 entre las primeras 5 de código abierto utilizadas para manejar facturas [5]. La arquitectura de SimpleInvoice está fundamentada en el modelo Cliente-Servidor basada en Web.

En este experimento se simula una situación donde un Software está instalado en un ambiente con recursos de Hardware muy limitados y se pretende mejorar el desempeño de 3 tareas principales que realiza: Presentar toda la lista de clientes en el sistema, presentar todos los productos y buscar un cliente en particular.

Métodos Analíticos

Las métricas utilizadas en este experimento, son comúnmente utilizadas para medir desempeño en comunicaciones de redes y ambientes Cliente Servidor. Estas métricas son ampliamente descritas en el libro "El Arte de Pruebas de Desempeño por "Ian Molyneaux" así como en una publicación del IEEE sobre Métricas de Desempeño a nivel de Cliente Servidor en arquitectura Modelo-Vista-Controlador (MVC por sus siglas en inglés) [6] [7].

Las métricas utilizadas fueron las siguientes:

- **Disponibilidad:** La cantidad de tiempo en que una aplicación está disponible para el usuario final. En términos de desempeño, esto significaría la incapacidad completa de un usuario para hacer uso efectivo de la aplicación; ya sea porque la misma simplemente no responde o el tiempo de respuesta se ha degradado hasta un grado inaceptable.
- **Tiempo de Respuesta:** La cantidad de tiempo que tarda la aplicación para responder a una petición del usuario. Este se refiere al tiempo desde que el Cliente solicita el servicio de la aplicación hasta que recibe su petición.
- **Desempeño:** ("Throughput"): La velocidad en la que ocurren los eventos orientados a la aplicación.
- **Utilización**: El porcentaje de la capacidad teórica de un recurso en uso. Esto incluye la cantidad de memoria y CPU utilizada por los recursos (Cliente, Servidor).
- **Utilización de la Red:** Data presentada sobre la red.

Figura 2
Representación Gráfica del Proceso IAO

Sometiendo el Software al Proceso IAO

El Software SimpleInvoice fue sometido a las tres fases propuestas del proceso IAO en la cual se documentaron todos los resultados de pruebas de rendimiento incluyendo soluciones aplicadas. La figura 2 representa los componentes de IAO.

Fase 1 - Inventario

La toma de inventario juega un papel muy significativo sirviendo como base para el resto del proceso. Situaciones o recursos no identificados pueden tener serias repercusiones en los resultados obtenidos. Los requerimientos funcionales en este experimento fueron los siguientes:

- Mejorar el desempeño de 3 tareas importantes que tiene el Software que relativamente requieren demasiado tiempo. Estas son: peticionar todos los clientes existentes en la Base de Datos que incluye total y balance por factura, indagar sobre todos los productos existentes en la base datos y buscar clientes específicos.
- Implementar seguridad de sistemas incluyendo medidas de protección para guardar datos sensitivos como son tarjetas de crédito o cuentas de pago.

Teniendo una lista de los requerimientos, se tomó un inventario de recursos de Hardware y Software. El mismo consistió de un laboratorio de pruebas diseñado para modelar la situación donde el Software esta implementado en un ambiente que presenta recursos muy limitados. Los componentes de este laboratorio de pruebas fueron los siguientes:

- **Hardware**: Se utilizaron 7 computadoras modelo Dell E6440, tipo laptop las cuales incluían un procesador Intel Core I5-4200 CPU @2.5GHZ ,2 Core(s), 4 logical Processor(s)-64bits, una memoria de 4.00gb, adaptador de Red modelo1506 802.11n (2.4ghz) y un sistema operativo Windows 7 Professional. También se utilizaron 2 computadoras de escritorio modelo HP DC5000 que incluían un procesador Pentium 4 CPU 2.80 GHz, una memoria de 1.00gb, un adaptador de Red tipo USB modelo

WNA3100 inalámbrico y un sistema operativo Windows XP 2002 SP 3. Como servidor, se utilizó una computadora marca Emachine modelo EL1300G que incluía un procesador AMD Athlon 2650e, 1600 MHz-64bits con una memoria de 3.00gb y un sistema operativo Windows 7 Professional.

- **Software**: El Software instalado en el servidor estuvo compuesto del programa SimpleInvoice utilizando XAMPP que contiene una distribución de Apache y permite integrar PHP y la Base de Datos MySQL. XAMPP Funciona como un Software de Servidor de Aplicaciones donde Apache opera como Servidor Web.
- **Red**: Router modelo NGB416N con puertos LAN/WAN 10/100 Base-TX (RJ-45) y velocidad de 150 Mbps.
- **Arquitectura**: Cliente-Servidor.

Fase 2 - Análisis

La segunda fase del proceso propuesto consistió en analizar todo el funcionamiento de la aplicación tal como está diseñada e implementada sometiéndola a una rigurosa auditoría de sistema. La auditoría de sistemas está dirigida a evaluar los métodos y procedimientos de uso en una entidad con el propósito de determinar si su diseño y aplicación son correctos [8]. De esta manera se comprueba el sistema de procesamiento y se identifican aspectos susceptibles a mejorarse o eliminarse.

Como parte del experimento, se sometió el Software a una serie de pruebas de rendimiento, específicamente sobre las funciones inventariadas en la fase anterior. El objetivo principal fue medir el rendimiento actual de estas funciones.

Para obtener resultados fiables se poblaron dos de las tablas principales de la Base de Datos con 100,005 tuples. Las tablas afectadas fueron "si_customers" y "si_products" las cuales mantienen los datos de clientes y productos.

Pruebas de Rendimiento en Cliente

En la primera prueba, se midió el tiempo de respuesta entre el cliente y el servidor cuando se utiliza la opción de seleccionar todos los clientes. Esta funcionalidad es muy utilizada en el sistema ya que para ver o editar un cliente, se tiene que usar esta opción. En esta prueba se utilizó el Navegador Internet Explorer 11. Este tiene una utilidad para medir rendimiento, depurar códigos y ayudar en el desarrollo de páginas llamada "F12 Developer Tools" la cual fue utilizada para medir el rendimiento del Software.

Utilizando la herramienta F12, se tomó una captura de los procesos que se ejecutaron al activar y utilizar la función de seleccionar todos los clientes. El programa mostró todos los datos en grupos de 25 indicando la cantidad total de datos (100,004) en la parte inferior. Accionar la función de solicitar todos los clientes y sus respectivos balances por factura, tomó 13,213 ms. Esta ejecución fue el resultado de 27 sub operaciones de las cuales 26 de ellas fueron métodos tipo GET y una de ellas tipo POST. Los métodos GET tuvieron un tiempo de procesamiento de 16,635 ms, promediando un tiempo por método de 639.80 ms con una desviación estándar de 374.91 ms. Las sub operaciones ocurrieron de forma concurrida. Por tal razón la suma de los tiempos de procesamiento en los métodos es mayor al tiempo total de cargar la página. Los métodos GET no fueron de mayor impacto en comparación con el método POST. Este método, importó los Clientes de la Base de Datos por medio de una petición tipo XML con una duración de 11,794 ms representando un 41% de toda la ejecución. El método POST fue el encargado de importar los datos que se cargaron en la página demorando 11.264 s. La figura 4 muestra los resultados obtenidos de la prueba donde se refleja que la función tomó 13,713 ms y un tiempo de procesamiento de 30,379 ms. El Servidor mantuvo comprometidos sus recursos provocando un cuello de botella que detuvo nuevas peticiones por 9.20s.

Se tomó una muestra de los recursos invertidos desde el lado del Cliente y la misma reflejó que la utilización del CPU fue mínima ocupando menos del 2% de los recursos. En términos de memoria, solo se incrementaron 1,383.72kb al completar la función. Los recursos de red tampoco estuvieron

comprometidos. El pico más alto reflejo la utilización de 0.29% por menos de 1 segundo.

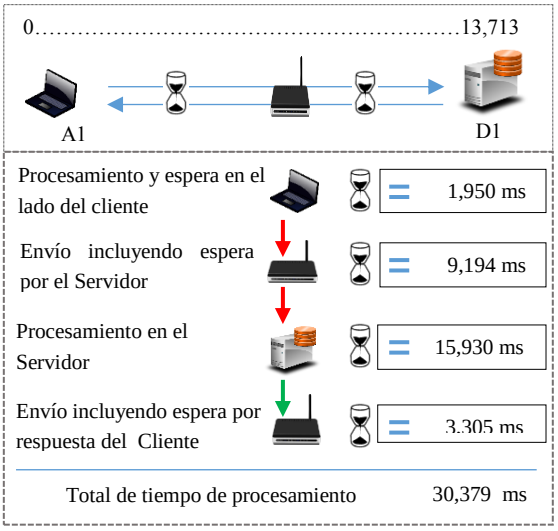

Figura 4
Tiempo de Procesamiento Prueba 1

En la segunda prueba se realizó una captura similar a la prueba 1 con la diferencia de que en esta ocasión se puso a prueba la función de obtener todos los productos existentes en el sistema. Los resultados de esta prueba fueron muy similares a la prueba anterior. El sistema utilizó la misma metodología de presentar los datos en grupos de 25 incluyendo el total de productos encontrados. Al igual que en la prueba 1 se iniciaron 27 métodos los cuales enviaron 12.96 kb. Peticionar todos los productos y cargar la página tomó 13,136 ms. Los recursos de memoria, CPU y red tuvieron similar utilización Al igual que en la prueba anterior el Método POST tuvo un impacto mayor, en comparación con el resto de las operaciones En esta prueba también se reflejó el cuello de botella que se presentó en la prueba 1. En esta ocasión el Servidor se mantuvo comprometido dejando peticiones en fila por 9,194 ms.

Para propósitos de medir consistencia en los resultados, se realizó una tercera prueba tomando una muestra de capturas de la función de indagar todos los clientes y sus respectivos balances por facturas tal como se realizó en la prueba 1. Los resultados de tiempos de procesamiento y tiempo transcurrido presentaron completa similitud con la prueba 1. El promedio de tiempo fue 13,410 ms con una desviación estándar de 477 ms. En términos de recursos de CPU, memoria, disco y utilización de red, los resultados fueron muy similares con muy poca utilidad de los mismos.

En una cuarta prueba se trajo un elemento adicional, concurrencia. En esta prueba se midió el tiempo de respuesta por el Servidor tomando 3 capturas por cada dispositivo al presionar el enlace "Customers" de forma simultánea produciendo múltiples instancias. A diferencia de las pruebas anteriores, las cuales presentaron resultados con una distribución simétrica, en esta ocasión estos fueron muy dispersos. El promedio de tiempo que antes se mantuvo en 13,312 ms se disparó un 70%. Algunas instancias reflejaron un tiempo de 13,666 ms, mientras que otras que tomaron 26,973 ms. Las diferencias se observaron en todos los eventos ("Wait", "Start", "Request", "Response"). Estas ocurren debido al efecto avalancha que provoca el retraso en el procesamiento en el lado del Servidor. Mientras el Servidor está manejando las peticiones se forma un cuello de botella que provoca detención de procesos en espera.

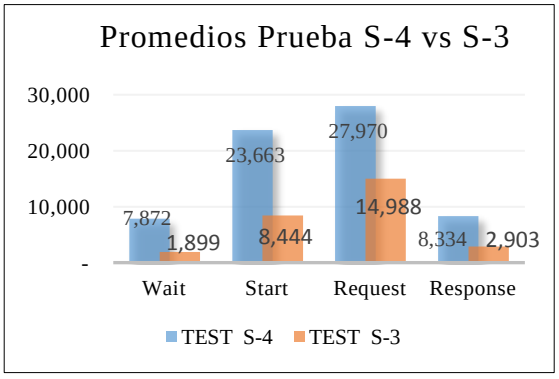

Figure 5
Comparable de Tiempos de Procesamiento entre Pruebas 4 y 3

En la figura 5, claramente se puede observar como el evento de espera ("Wait") incrementó un 315% en comparación a las pruebas S-3 donde acceder al enlace "Customer" ocurrió de manera secuencial. Las múltiples peticiones al Servidor mantuvieron diferentes eventos en fila los cuales se reflejan marcadamente en los eventos de espera e inicio. Durante el evento Inicio ("Start"), múltiples sub operaciones no pudieron ser instanciadas a la

espera del Servidor. Evidentemente, este estuvo comprometido por lo que no pudo responder de forma inmediata a las peticiones de los clientes.

Conclusión de Pruebas del Lado del Cliente

Los datos de las pruebas sometidas revelan que el tiempo de procesamiento en el lado del Servidor es un factor crítico en el tiempo de respuesta. Un tiempo de respuesta de aproximadamente 23 segundos definitivamente puede ser percibido como uno insatisfactorio. Claramente, los recursos de Servidor quedan comprometidos durante el evento que solicitan los Clientes. Este tiempo de respuesta en un ambiente de 10 o más usuarios interactuando incrementaría el mismo hasta un nivel de gravedad que pudiera hacer colapsar el Servidor. Mejorar este tiempo de procesamiento es un factor clave para mejorar desempeño.

¿Puede un Navegador Distinto Mejorar Desempeño?

Bajo la prerrogativa de si un navegador pudiera presentar mejores tiempos de respuesta, se replicaron las pruebas S1-4 donde múltiples estaciones realizaban peticiones al Servidor pero en esta ocasión con el navegador Google Chrome.

Al igual que el navegador Internet Explorer, Chrome tiene una herramienta de múltiples utilidades para medir desempeño en la red, recursos, depuración e incluso identificar vulnerabilidades. Esta herramienta fue utilizada para tomar 10 capturas al seleccionar la opción de solicitar todos los clientes existentes en el sistema. Según los datos obtenidos, Google Chrome muestra menores tiempos de procesamiento que IE lo que aparenta demostrar una reducción de un 30% de tiempo de procesamiento. Sin embargo, al comparar el tiempo de respuesta, Google Chrome logró superar a IE solamente por un 1% en el promedio de tiempo. Según los datos, el navegador Chrome muestra un mejor tiempo en descargar datos del Servidor luego de que este inicie el envío. Este promedio se redujo por 995 ms en comparación con los 3,066 ms que reflejó IE. El tiempo de procesamiento en el lado del Servidor, aparenta mostrar una pequeña mejoría de 4.14%. Estas pruebas reflejan que el tiempo de respuesta del Servidor es el factor más dominante de toda la operación al igual que demostraron las pruebas utilizando IE11.

En términos de utilización de CPU cabe mencionar que los resultados de las capturas muestran mayor utilización en Google Chrome que en IE. Chrome llegó a ocupar hasta un 12% de los recursos de CPU mientras que IE mantuvo menos de 2% durante la ejecución. La memoria utilizada por Chrome al cargar la página ocupó 7.2mb en comparación con IE que solo requirió 2.04mb.

En conclusión, Google Chrome demostró ser uno 1% más rápido en comparación con IE a un costo de mayores recursos utilizados.

Analizando el Servidor

Las pruebas del lado Servidor consistieron en el monitoreo de recursos mientras el mismo interactúa con las peticiones de los Clientes replicando las pruebas 1 y 2. En las pruebas anteriores, los Clientes interactuaron con el Servidor al peticionar los Clientes cuando presionaban el enlace “Customer”. En esta ocasión se documentaron los efectos de esas operaciones en el lado del Servidor.

Como se había mencionado anteriormente, el equipo utilizado como Servidor utiliza diferentes programas para manejar la Base de Datos, el Servidor Web y otras aplicaciones. En adición, el Sistema Operativo maneja distintos servicios como un Firewall y Microsoft Security, los cuales tiene un costo de utilización. Tomando una muestra de los recursos comprometidos por el servidor al arrancar, se observó que estos servicios requieren un 41% de memoria y una fluctuación de entre 5% y 9% del CPU.

Durante la operación de responder a la petición del Cliente, el Servidor se mantuvo sumamente comprometido utilizando un 100% de todos sus recursos. En las pruebas anteriores se mencionó que el impacto mayor en términos de tiempo de procesamiento y tiempo de respuesta era responder al Método POST que accionaba una petición a la base de Datos. En efecto, responder a esa petición agotó los recursos antes mencionados. Explorando

los servicios que se accionaron en el lado del Servidor, cuando se inició la petición, se observó que los procesos mysqld.exe y httpd.exe comenzaron a incrementar los usos de CPU a tal nivel que el procesador se mantuvo ocupado al 100% hasta terminar la ejecución. Definitivamente a un servidor tan comprometido le resultaría imposible atender nuevas peticiones. El proceso que mayores recursos de CPU ocupó durante toda la actividad fue mysqld.exe. Este es un servicio que se procesa en segundo plano, el cual es utilizado por la Base de Datos MySQL.

En términos de memoria, completar la operación ocupó aproximadamente 5mb. En cuanto a utilización de Red no hubo grandes compromisos manteniendo los niveles por debajo de 1%.

Cuello de Botella Identificado

Los procesos internos que manejó el Servidor, mientras interactuaba con la base de datos, agotaron los recursos provocando un cuello de botella que puso en fila diversas operaciones. Múltiples factores pueden ocasionar cuellos de botella pero los 2 componentes más influentes son [9]:

- **CPU**: La capacidad de procesamiento del DBMS debe coincidir con la carga de trabajo prevista del sistema. Una alta utilización de CPU podría indicar que la velocidad del procesador es demasiado lenta para la cantidad de trabajo realizado. Un cuello de botella en el CPU no tan solo afecta el DBMS sino todos los procesos que se ejecutan en el sistema.
- **Códigos en la Aplicación:** No todos los cuellos de botella son causados por los recursos de Hardware limitados. Una de las situaciones más comunes ocurre como resultado de códigos de programación mal escritos.

Tomando en consideración los dos factores mencionados y habiendo identificado que los procesos de la base de datos comprometieron seriamente el Servidor, se realizó un análisis de la Base de Datos, específicamente las tablas afectadas cuando se utiliza la opción de solicitar todos los productos o todos los clientes. Uno de los principales factores que provoca pobre desempeño en la base de datos ocurre por razones de débil o ninguna indexación y mal diseño de las consultas [10]. Indexar una base de datos requiere de una buena comprensión de los datos y consultas. Los índices utilizan componentes clave de los datos de una tabla en una estructura binaria para mejorar la capacidad de búsqueda. Cada registro de datos en la tabla debe estar asociado con datos en el índice. Indexar puede aumentar notablemente la velocidad de búsqueda.

Partiendo de esa premisa se procedió con el análisis de las consultas y se descubrió que las tablas estaban parcialmente indexadas por lo que se realizaron pruebas de rendimiento de estos Queries con las utilidades de MYSQL. Los resultados revelaron que los índices que contenían las tablas no guardaban ninguna relación con las consultas de productos y clientes.

Resultados Fase de Análisis

Por medio de la fase 2 se descubrieron los factores que causaban pobre desempeño en las funciones de obtener clientes y productos. Los hallazgos dan lugar a un plan de optimización que va relacionado a corregir el pobre desempeño de las consultas a la base de datos.

Fase 3 de Diseño y Optimización

Ya identificadas y documentadas las deficiencias del Software es momento de ejecutar el plan de acción para mejorar el desempeño.

Mejorando los índices de las tablas afectadas en la consulta de productos logró reducirse el tiempo de ejecución en un 24% y la cantidad de filas que originalmente tenía que leer las Base de datos.

Indagando el tipo de motor de almacenamiento de la base de datos se observó que la tabla utiliza MYISAM. La versión más reciente de MySQL contiene un nuevo motor de almacenamiento llamado Aria que promete ser una alternativa a MyISAM con mejor desempeño. Se procedió a cambiar el mecanismo y se consiguió una mejora de un 28% al compararla con la tabla indexada. La combinación de mejorar los índices y cambiar el mecanismo de almacenamiento redujo en un 45% el

tiempo de ejecución. Finalmente, se analizó el Query que utiliza el Software para consultar los Productos y se identificaron deficiencias, las cuales dieron lugar a optimizar el Query substituyendo las clausulas WHERE por NATURAL JOIN lo que produjo una reducción de un 33%. Básicamente, se refractó el Query eliminando la correlación de las sub consultas encapsulando las selecciones con el uso de NATURAL JOIN para totalizar las mismas. La idea fue tratar las sub consultas como tablas virtuales y unirlas con la tabla "si_products" Los resultados de la optimización se muestran en la figura 6.

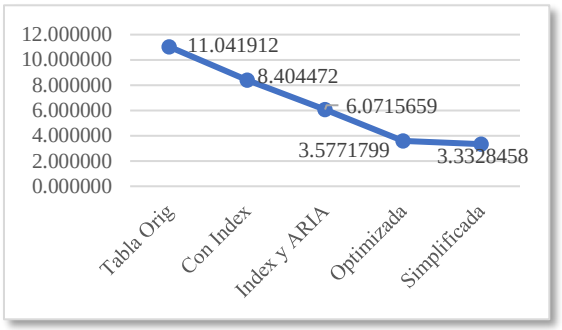

Figure 6
Reducción de Tiempo de Ejecución como Resultado de Mejoras

Como se puede apreciar en la figura 6 el tiempo promedio que tomaba la consulta original fue reducida a 3.33 s representando una reducción de un 70%. Se aplicó la misma solución a la función de clientes y se redujo la misma a un 61%.

IMPLEMENTANDO NUEVAS TECNOLOGÍAS

Habiendo mejorado el desempeño de las funciones identificadas en el alcance de este proyecto más de un 60%, es momento de identificar nuevas tecnologías que puedan mejorar el desempeño aún más. La fase de Diseño y Optimización debe contemplar el nivel máximo de optimización. Esto incluye, no tan solo identificar deficiencias en el diseño sino también considerar el uso de nuevas tecnologías que puedan favorecer la aplicación.

Arquitectura SPA

Mediante la exploración de diferentes tecnologías que pudiesen llevar la aplicación a un nivel superior se identificó el concepto SPA ("Single Page Application" por sus siglas en ingles). Un Software de página única (SPA) o aplicación de página única, es una Aplicación o sitio web que cabe en una sola página, con el propósito de dar una experiencia más fluida a los usuarios como lo hace un Software de escritorio (Desktop) [11]. En un SPA todos los códigos de HTML, JavaScript, y CSS se cargan de una vez y los recursos necesarios dinámicamente como lo requiera la página. Estos se van agregando, normalmente como respuesta de las acciones del usuario. La página no tiene que cargar otra vez en ningún punto del proceso. Tampoco se transfiere a otra página, aunque las tecnologías modernas (como el pushState () API del HTML5) pueden permitir la navegabilidad en páginas lógicas dentro de la aplicación. La interacción con las aplicaciones de página única pueden involucrar comunicaciones dinámicas con el Servidor web que está detrás. Un SPA se mueve lógicamente del Servidor al Cliente. Este cambio arquitectónico ha sido llamado como "Arquitectura de Servidor Delgado" para remarcar que la complejidad ha sido trasladada del Servidor a los Clientes.

En las aplicaciones Web tradicionales, cada vez que es necesario cargar una página, el Cliente llama al Servidor, éste obtiene el HTML necesario, lo envía al navegador y éste se refresca para mostrar los nuevos datos. En contraposición, un ambiente que implementa SPA, solicita el contenido en una primera llamada y no vuelve a interactuar directamente con el Servidor hasta que surja una necesidad de datos adicionales. A partir de ese momento todas las interacciones con el servidor se producen mediante llamadas AJAX, las cuales devuelven los datos necesarios. El navegador no necesita refrescar la página, sino que está lo hace dinámicamente con los datos solicitados. La figura 7 muestra una representación grafica del ciclo de una página SPA.

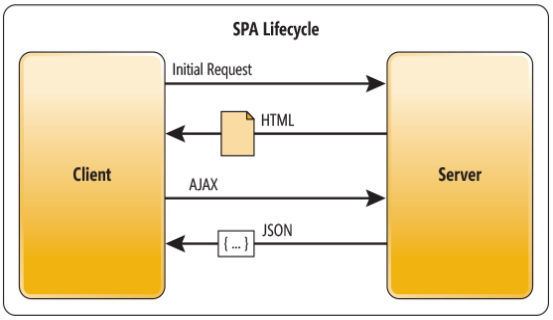

Figure 7
Ciclo de Vida de una Página que Implementa SPA

Este tipo de arquitectura es muy adecuada para aplicaciones como la presentada en este proyecto donde se reducirían grandemente las peticiones al Servidor.

Implementando SPA en SimpleInvoice

Anteriormente se identificó que acceder a cualquiera de los dos enlaces ("Customers ", "Products") requería de 27 cargas que promediaban 13.71s. Con la optimización anterior se redujo el tiempo a un promedio de 6.7 segundos pero la cantidad de descargas no fue disminuida. Para reducir esa cantidad de eventos, y poder mejorar aún más el desempeño, se diseñó un plan de optimización para utilizar la arquitectura SPA en SimpleInvoice.

Cuando los usuarios acceden los enlaces que presentan los clientes o productos, el sistema carga todos los datos existente de las respectivas tablas. Si el usuario necesita buscar un cliente especifico, el sistema tiene que invocar una solicitud de datos. Esta acción elimina el contenido en Buffer y carga la página con el resultado. Si no se encontraron records, presionar el enlace tiene un costo de una nueva petición para cargar los archivos. Lo que parecería una opción de retroceder tomaría aproximadamente 15.00 s

Cambiando la arquitectura de SimpleInvoice al concepto SPA, la página no tendría que peticionar nuevamente los datos ya que se iniciaría una búsqueda de los datos en el Buffer lo que sería mucho más rápido y efectivo. La mayor parte del contenido Web de SimpleInvoice se genera dinámicamente por medio de códigos de JavaScript en diversos marcos como lo son Ajax, JQuery y Json así como también códigos PHP. Para incluir las funciones de AngularJS se modificó el código PHP que generaba HTML de la página "Customer y se incluyeron utilizando las librerías de AngularJS. Se rediseño el algoritmo de obtención de datos manejando la data importada del XML almacenándola en un arreglo. Una vez conservados los datos en el arreglo, las directivas de angular conservan el contenido en memoria permitiendo al usuario poder realizar las búsquedas en el contenido sin tener que reiniciar una petición adicional al Servidor.

Los resultados de esta implementación fueron sumamente exitosos según se muestran en la figura 8. Las búsquedas ocurren extremadamente rápidas tomando menos de 1 segundo en buscar y recuperar los datos en pantalla.

Optimizar la funcionalidad de buscar los Clientes en la aplicación no implicó eliminar los objetos existentes sino que más bien se incorporaron las funciones dentro del contenido HTML y códigos de JavaScript.

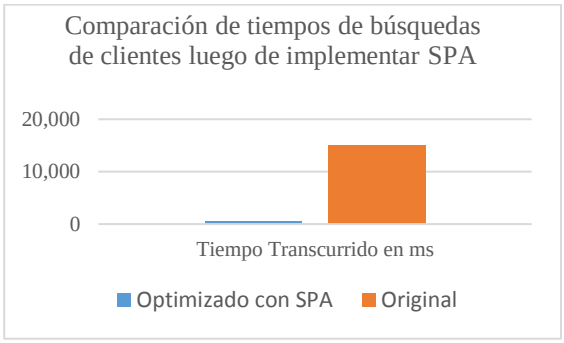

Figure 8
Búsqueda de Clientes antes y después de SPA

Seguridad a Nivel de Base de Datos

Después de haber completado los procesos de inventario, análisis y optimización ahora podemos hablar de seguridad de sistemas. Las condiciones en las que operaba la aplicación dificultaban la implementación de medidas de seguridad ya que los costos de utilización eran demasiado altos para los recursos disponibles.

Observando la estructura de tablas se identificó lo siguiente:

- **No existe Log**- La estructura de la Base de Datos de SimpleInvoice no cuenta con tablas que resguarden información sobre las entradas, eliminaciones, ediciones o inclusiones de records.
- **No Cifrado**- La información sensitiva como tarjetas de crédito no está encriptada.

Mejoras en la Seguridad

La estructura diseñada para mejorar la seguridad de la Base de Datos sin afectar el desempeño de la aplicación fue manejada en el orden de puntos críticos encontrados durante la etapa de recopilación. Estos fueron diseñados de la siguiente forma:

- **No Log-** Es sumamente necesario mantener un log de las principales actividades que llevan a cabo los usuarios especialmente en el registro de entradas a la aplicación, creación de cuentas, pagos, facturas, y otros. Estos Logs no solo ayudan a monitorear actividades sino que en casos de ataques se puede contar con una bitácora que ayude durante el proceso de investigación forense. Para mejorar esta vulnerabilidad se crearon tablas tipo Log para registrar datos referentes a entradas, ediciones o alteraciones. Estas tablas fueron atadas a las principales tablas del sistema utilizando la opción de "Triggers". El mecanismo de "Triggers" elimina la necesidad de insertar a estas tablas por medios de comandos en el Software. Si un atacante altera de cualquier manera los ficheros PHP todavía quedaría una capa de seguridad a nivel de Base de Datos que mantendría registros de actividad. En términos de desempeño, es más conveniente dejar estas ejecuciones como operaciones internas de la Base de Datos.
- **No Cifrado**- La medida propuesta para estos fines es el uso de encriptación de MYSQL para todas las columnas requeridas. El uso de la encriptación AES en MySQL es sencillo, y se realiza mediante dos funciones de encriptación: AES_ENCRYPT y AES_DECRYPT. Estas funciones permiten usar AES_128 o AES_256. Esta medida sería excelente para guardar información sensitiva en la base de datos como lo pueden ser números de tarjetas de crédito de clientes entre otros. Se modificaron los ficheros PHP que contenían el código de almacenar información de pagos y se incluyó la función AES_ENCRIPT para guardar los datos.

DISCUSIÓN

Los resultados obtenidos demuestran como se pudo mejorar el rendimiento de una aplicación sin la necesidad de remplazar Hardware o Software al aplicar la metodología de procesos IAO. Un inventario de recursos y requerimientos preciso, en combinación con un análisis completo, fueron factores claves para identificar deficiencias. De otro lado, mejoras al diseño original del Software en conjunto con la integración de nuevas tecnologías, permitieron completar una optimización a un alto nivel.

TRABAJO FUTURO

En este trabajo se incluyeron una serie de técnicas para mejorar el rendimiento de un Software implementado en un ambiente con pobres recursos de Hardware. Entre las técnicas aplicadas estuvieron la corrección de deficiencias en el diseño de la base de datos, comparación de diferentes navegadores y la aplicación del concepto SPA. Sin embargo, quedó pendiente experimentar la implementación del Software en diferentes sistemas operativos.

De otro lado, los resultados obtenidos al incluir los módulos y directivas de AngularJS fueron muy favorables. Existen otros marcos de JavaScript que han adoptado los principios de SPA. Uno de estos es ReactJS el cual es patrocinado por compañías como Facebook e Instagram [12]. ReactJS genera un DOM virtual para cada componente creado, al cual aplica un algoritmo que compara los elementos. Con esta utilidad, se identifican posibles cambios dentro el DOM Virtual y solo se aplican los mismos para evitar repintar todo el DOM.

La técnica de complementar el uso de ReactJS con AngularJS podría traer resultados de rendimiento aún mejor ya que no tan solo se mejoraría el manejo de datos sino que también la forma en que se pinta el DOM.

CONCLUSIÓN

Un proyecto de optimización de un Software puede ser una tarea sumamente compleja. En muchas ocasiones, los requerimientos funcionales son mayores al alcance que puede ser cubierto en un proyecto de esta índole. Las etapas de Inventario y Análisis son fases claves para completar las expectativas a tiempo y concordancia con el presupuesto. No seguir una metodología de procesos adecuada puede traer serias implicaciones como intentar solucionar demasiadas situaciones que van por encima de los recursos dispuestos.

De acuerdo con los datos presentados en este trabajo, el proceso IAO probó ser un instrumento efectivo para manejar situaciones como las presentadas en este documento.

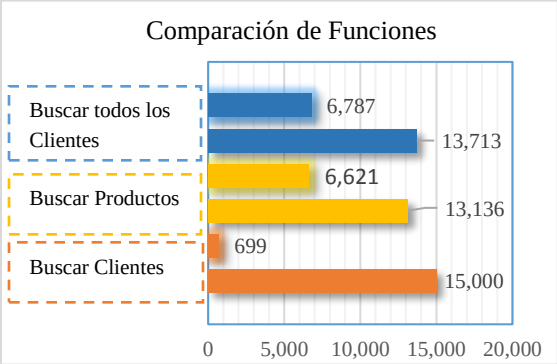

Figure 9
Comparación de Funciones Optimizadas

La figura 9 muestra una comparación final del mejoramiento en tiempo de ejecución sobre las funciones optimizadas.

REFERENCIAS

[1] G. Antunes, F. Freitas, J. Borbinha, "Designing Digital Preservation Solutions: A Risk Management-Based Approach," in the *International Journal of Digital Curation*, vol. 5, Issue 1, 2010, pp. 5-11.

[2] Statista, "Global Enterprise Application Software Revenue in 2011 and 2016, by segment," in *Article*. Available: http://www.statista.com/statistics/247549/global-enterprise-application-software-revenue-by-segment.

[3] Panorama Consulting, *2015 ERP Report*, 2015, pp.1-4.

[4] S. Radack, "Systems Development Life Cycle (SDLC)," in *NIST Special Publication (SP) 800-64*, 2008, pp.11-38.

[5] The Hosting News. 5 Free and Open Source Invoicing Systems [Online] Available: http://www.thehostingnews .com/5-free-and-open-source-invoicing-systems.html.

[6] I. Molyneaux, *The Art of Application Performance testing*, 1st ed., O' Reilly Media, Inc., ch.1-5, pp. 3-127.

[7] J. Badgujar, M. Jailia and A. Kumar, "Performance Metrics of Web Crawler in Client-Server and MVC Architecture" in *IEEE*, INSPEC Accession Number 15309881, pp. 2-10.

[8] Technical Committee ISO/TC 176, ISO 19011:2011, "Quality management and quality assurance", 2011, pp.4-24.

[9] P. Rob and C. Coronel, *Database Systems Design, Implementation and Management*, 8th ed., Cengage Learning, 2009, ch. 11, pp. 450-487.

[10] A. Pande and F. Charvet, "Database Performance Study," in *Study at University of UMSL*, pp.4-52.

[11] M. Galli, R. Soares, "Inner-browsing Extending the Browser Navigation Paradigm", in *Netscape Communications*, 2012. Available: https://developer.mozilla.org/en-US/docs/Archive/Inner-browsing_extending_the_browser_navigation_paradigm.

[12] A. Fedosejev, *React.js Essentials*, Packt Publishing, 2015, ch. 2, pp.17-36.