Integración Big Data en el Mundo empresarial: ELK - Hadoop - Splunk

Aitor Santacreu Grifol

Resumen— Hoy en día todos los procesos automatizados generan *logs* de sistema para poder ser auditados, y en consecuencia la volumetría de datos que se generan puede ser un problema a la hora de realizar algún tipo de análisis en el ámbito del Big Data. A partir de este artículo se detalla en qué consisten las herramientas de indexación y visualización de datos en Big Data. En concreto, se propone el análisis de dos herramientas de este tipo, una de gratuita, *stack* ELK, y otra de pago, Splunk. El análisis de estas herramientas se realiza a partir de un caso de uso provisto por una organización privada, desplegando y adaptando cada arquitectura. El resultado de este trabajo es establecer un marco comparativo entre ambas herramientas para responder a las preguntas planteadas por el caso de uso. También se determina que herramienta debe usarse en otros casos de uso.

Palabras clave— Big Data, Splunk, Elasticsearch, Logstash, Kibana, *stack* ELK, Hadoop, Cloudera, Amazon Web Services, computación en la nube, herramientas de indexación.

Abstract— Nowadays all automated processes generate system logs to allow audits. The volume of the generated data can become a problem when doing some kind of analysis in Big Data environments. This article details what they are indexing and visualization tools in Big Data. Concretely, it is proposed an analysis of two tools, one free to use, stack ELK, and another of payment, Splunk. The analysis of this tools will be done based on a use case, provided by a private organization, both tools will be deployed and adapted. The result of this project is to establish a comparative frame between both tools to answer the questions made by the use case. Also determinate which tool must be used in other use cases

Index Terms— Big Data, Splunk, Elasticsearch, Logstash, Kibana, *stack* ELK, Hadoop, Cloudera, Amazon Web Services, cloud computing, indexing tools.

1 Introducción

La constante evolución del hardware y software durante las últimas décadas ha desbocado en la denominada *Revolución Tecnológica*, con un gran impacto en la sociedad. Actualmente la media de dispositivos con conectividad a Internet por persona es de 3.64 según *Global Web Index* [1]. La consecuencia directa del aumento de dispositivos y prestaciones de los mismos es el crecimiento exponencial de datos, provenientes de fuentes muy diversas, como: *logs* de sistemas, sensores, redes sociales… Incluso los mismos datos generan metadatos (hora de generación, lugar, dispositivo generador…).

Para afrontar esta inmensa cantidad de información surge el Big Data. Las soluciones Big Data han sido desarrolladas para satisfacer las necesidades empresariales, tales como almacenar, procesar, obtener valor analítico y visualizar el conocimiento generado a partir de los datos del *Internet de las cosas*. Los datos sin propósito son solo eso, datos, gracias al Big Data, se convierten en una ventaja competitiva. Según Forbes y Gartner, el 83% de las grandes compañías invierten en Big Data y obtienen un beneficio directo de ello [2]. Por ejemplo, algunas compañías lo usan como pilar fundamental: Facebook, Google, Amazon, eBay fueron construidas con finalidades Big Data, y su beneficio depende de ello, al recopilar datos y metadatos de los usuarios y utilizarlos con finalidades comerciales (publicidad dirigida). Además de las ventajas respecto al análisis de la información, las arquitecturas Big Data tienen un coste menor, ya que se aprovechan las ventajas de la computación distribuida. Por ejemplo, almacenar 1TB de datos durante un año con bases de datos tradicionales tiene un coste aproximado de $37,000[3], almacenarlo en un clúster de Hadoop costaría $2,000. Las reducciones de coste y el incremento de los beneficios, fomentan el uso de las tecnologías Big Data.

Este artículo sigue la siguiente estructura:

La **sección 2** presenta el contexto del proyecto a través del **estado del arte**. La **sección 3** muestra los **objetivos** a alcanzar. La **sección 4** expone el **caso de uso** práctico. La **sección 5** presenta la **metodología** seguida. La **sección 6** muestra los **resultados** obtenidos. La **sección 7** presenta las **conclusiones** que se han podido extraer. La **sección 8** engloba las **conclusiones personales**.

- *E-mail de contacte: aitor.santacreu@e-campus.uab.cat*
- *Mención realizada: Ingeniería del Software.*
- *Trabajo tutorizado por: Katerine Diaz (CVC)*
- *Curs 2015/16*

2 ESTADO DEL ARTE

La revolución tecnológica empezó hace más de 20 años, y ha causado una gran acumulación de datos por parte de las grandes empresas. Cada día se generan aproximadamente **2.5 hexabytes** de datos, con fuentes generadoras cada vez más variadas.

Uno de los mayores retos que se enfrentan las grandes corporaciones es el análisis de los eventos que generan los sistemas, ya sea para visualizar el estado de los mismos o bien para realizar auditorías. Los eventos de sistema son comúnmente llamados *logs* e incluyen mucha información. Estos normalmente se generan de un modo estructurado o semi-estructurado y provienen de cualquier tipo de dispositivo, programa o aplicación. Los retos son diversos: almacenar, analizar, consultar y visualizar los eventos, de un modo rápido y fiable.

Las herramientas de indexación han sido desarrolladas para afrontar estos retos, ya que en lugar de buscar la información directamente en el texto, se busca en un índice, siendo mucho más rápido. Hoy en día este tipo de herramienta es cada vez más común en entornos Big Data, y los datos que pueden tratar tienen una característica particular: han de ser datos basados en series temporales. A la hora de elegir una herramienta de este tipo, las empresas se enfrentan a un dilema, adaptar una arquitectura *open-source* o adquirir una arquitectura de pago:

- **Software *open-source***: productos de código abierto que permiten adaptarse a las necesidades a partir de programar nuevas funcionalidades desde el código desarrollado.
- **Software de pago**: productos de código propietario, que conlleva el pago de licencia de uso asociada al producto.

Escoger correctamente la herramienta en un proyecto es una de las decisiones más difíciles, ya que de ello dependerá el éxito o el fracaso del mismo. La elección puede depender de muchos factores:

- **Factores relativos al marco del proyecto**: presupuesto, personal, fecha límite, volumen de datos, confidencialidad…
- **Factores relativos a la tecnología**: velocidad de indexación, capacidad de adaptación, configuración…

La decisión por lo tanto no es trivial, ya que existen muchos productos, tanto de pago como gratuitos, y cada uno de ellos ofrece funcionalidades similares. Se propone la resolución de un caso de uso (**sección 4)** utilizando una arquitectura *open-source* y otra de pago. Las herramientas que se han seleccionado para desarrollar el caso de uso, son las que más se utilizan en el mundo empresarial:

- **Splunk**: herramienta de pago para indexar, analizar y visualizar datos.
- **ELK**: conjunto de herramientas open-source para indexar, analizar y visualizar datos.

En la **sección 5** se muestran en más detalle.

3 OBJETIVOS

Para analizar las herramientas de indexación, Splunk y ELK, se valorará su capacidad para interactuar con Apache Hadoop, que es la arquitectura más común para almacenar datos en entornos Big Data[4].

La Tabla 1 resume los objetivos del proyecto.

	Objetivos del proyecto
1	Montar un clúster funcional desplegando el ecosistema de Apache Hadoop usando una distribución de Cloudera Manager
2	Montar un servidor funcional utilizando Splunk donde se puedan indexar, analizar y visualizar los datos.
3	Montar un servidor funcional utilizando ELK donde se puedan indexar, analizar y visualizar los datos.
4	Establecer conexión entre ELK y el clúster con Hadoop.
5	Establecer conexión entre Splunk y el clúster Hadoop
6	Implementar el caso de uso en ELK y en Splunk.
7	Analizar los datos procedentes del caso de uso para obtener métricas útiles mediante ELK y Splunk.

Tabla 1. Resumen de los objetivos.

Los objetivos planteados buscan dar respuesta a las siguientes cuestiones:

	Cuestión
1	¿Cuál de las dos arquitecturas obtiene un mejor rendimiento, respecto a la velocidad de indexación?
2	¿Cuáles son las limitaciones de cada una de las arquitecturas?
3	¿Cuál sería el escenario óptimo para usar cada arquitectura?
4	¿Dado un caso de uso concreto, qué arquitectura se adapta mejor y con más facilidad?
5	¿Es más rápido desplegar una arquitectura *open-source* o una de pago?

Tabla 2. Cuestiones a responder a partir del desarrollo del proyecto.

4 CASO DE USO

Para poder establecer las bases que faciliten la elección de qué arquitectura de indexación es mejor, un cliente de Everis España ha proporcionado el caso de uso que se va a resolver.

El proveedor del caso de uso está haciendo pruebas con tres servidores en un entorno de pre-producción. Sus empleados interaccionan con un sistema que les permite gestionar los servicios que provee dicha empresa. Esta interacción a nivel interno lanza consultas hacia una base de datos relacional, o bien ejecuta funciones; uno de los servidores es un balanceador de carga que dirige las tareas hacia un servidor u otro, tal como presenta la Figura 1.

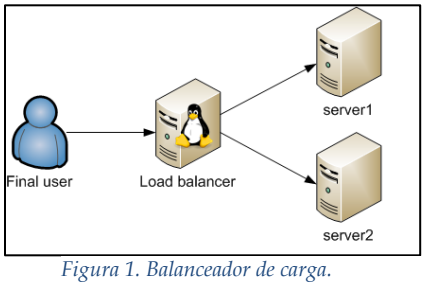

Figura 1. Balanceador de carga.

El análisis del rendimiento de los servicios que se ejecutan en los servidores y la verificación del funcionamiento del balanceador de carga, se realizará a partir de los eventos de sistema. Estos eventos quedan registrados en los servidores que ejecutan las tareas.

Los dos servidores finales, mostrados en la Figura 1, generan y almacenan en ficheros los eventos de sistema, que incluyen la actividad de los usuarios y las llamadas a funciones y procedimientos. Los ficheros proporcionados pesan cerca de 10GB (unos 50 millones de registros), y abarcan los evento de los dos últimos años.

A partir de la cesión de estos ficheros se analizará la capacidad que tienen Splunk y ELK para gestionar estos eventos, explotarlos y visualizarlos en tiempo real.

5 METODOLOGÍA

Este apartado da una visión global de las tecnologías desplegadas y las fases de desarrollo del proyecto.

5.1 Apache Hadoop

Cloudera Manager es una distribución *open-source* que provee todo el ecosistema de Apache Hadoop en un mismo entorno. Permite visualizar el estado de los servicios, sacar métricas de rendimiento del servidor… Todo ello mediante una interfaz gráfica.

Se desplegará Cloudera Manager para utilizar el sistema HDFS (Hadoop Distributed File System) [5]. Mediante HDFS se establece un factor de replicación para los datos, que permite al clúster ser resistente a fallos hardware; cada nodo de datos (*datanode*) almacena una parte de los datos, por lo que cuando un nodo de datos falla, los datos siguen estando en otros servidores, siendo accesibles en todo momento.

El factor de replicación más común es de tres por lo que se crearán tres nodos de datos. Todo clúster Hadoop precisa de un nodo especial que controla dónde se encuentran los datos (almacena metadatos en lugar de datos), dicho nodo se llama *namenode*. Como mínimo se necesita un nodo de este tipo, siendo necesario más de uno en entornos de alta disponibilidad. Si el *namenode* no se encuentra disponible, no se podrán acceder a los datos. Ya que la implementación no se enfoca hacia un entorno de alta disponibilidad, sino que se enfoca en un entorno de pruebas, no se precisa de un *namenode* de refuerzo.

Se desplegarán cuatro servidores que conformarán el clúster Hadoop (tres *datanodes* y un *namenode*).

5.2 *Stack* ELK

Siglas de *Elasticsearch*, *Logstash*, *Kibana*; tres herramientas que trabajan conjuntamente formando un flujo de trabajo[6]. Son herramientas *open-source* actualizadas, mejoradas y mantenidas por la comunidad de usuarios. *ELK* está diseñado para gestionar, explorar, analizar y visualizar datos procedentes de cualquier fuente.

En este proyecto se analiza la capacidad de adaptar este conjunto de herramientas a un caso de uso real, teniendo en cuenta las ventajas y desventajas que ello conlleva.

Elasticsearch es una herramienta gratuita (*open-source*) que indexa datos, un índice es el símil a una base de datos en un sistema relacional de bases de datos. Contiene el mapeo que define los múltiples tipos, dicho mapeo funciona como un esquema de definición en una base de datos relacional, así como las características de cada índice.

Logstash habitualmente se utiliza para dar formato a los datos que van a ser indexados en Elasticsearch (o en otros *outputs*). Para ello se configura Logstash mediante el uso de filtros y *plugins*.

Una vez indexados los datos en Elasticsearch, se configura **Kibana** para leer los eventos indexados y poder así consultarlos, analizarlos y visualizarlos mediante el uso de *Dashboards*. Kibana se usa mediante el navegador web, dispone de una interfaz intuitiva, permite ejecutar consultas contra los datos almacenados en Elasticsearch para obtener resultados y así poder crear las visualizaciones.

La Figura 2 muestra el flujo de datos habitual entre Elasticsearch, Logstash y Kibana. Logstash monitoriza un *input* y formatea el *output*, se redirigen los datos estructurados hacia Elasticsearch, una vez indexados se pueden visualizar mediante Kibana.

Figura 2. Flujo de datos ELK.

5.3 Splunk

Splunk es una de las herramientas de indexación líder en el mercado [7], a diferencia de ELK esta tecnología no es gratuita ni open-source. Al ser de pago, a priori, es una herramienta más completa, pero las funcionalidades principales son las mismas que ELK, permite gestionar, explorar, analizar y visualizar datos. Por esto las dos herramientas son comparables, porqué en principio realizan las mismas funciones, pero de un modo distinto.

5.4 Desarrollo

La metodología seguida para lograr los objetivos propuestos está basada en un modelo en cascada, que define las fases a partir de un proceso de descomposición del trabajo (EDT)[8]. La Figura 3 muestra las fases de este proyecto.

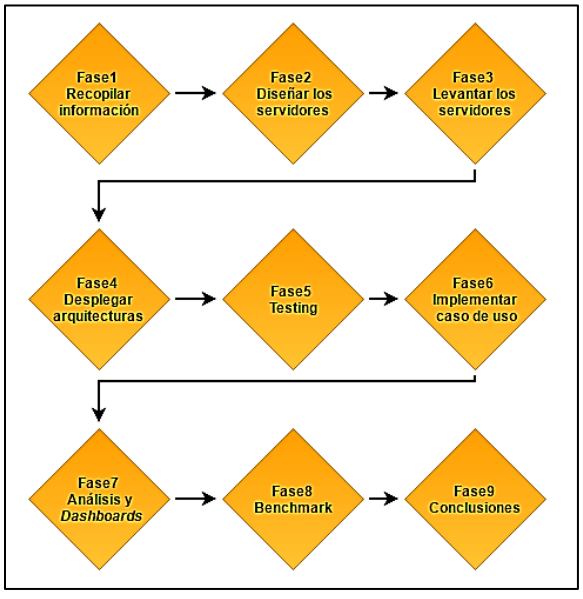

Figura 3. Metodología.

A continuación, se resume brevemente las principales fases del proyecto:

Fase 1 – Recopilación de información: en esta fase se definió el enfoque y el alcance a partir de un proceso documental exhaustivo, con la finalidad de tener una visión global del funcionamiento y de cada una de las tecnologías.

Fase 2 – Diseño de la arquitectura de los servidores: en esta fase se analizaron las ventajas que representa usar servidores virtuales en la nube en lugar de usar servidores físicos, respecto a coste, conectividad, escalado, control y recursos. En la Figura 4 se muestra el resultado de la comparativa entre los tres modelos principales de servidores: Cloud, Cloud *on-premise* y *on-premise*.

Se exploraron los servicios que ofrece Amazon en cuanto a Cloud Computing[9], ya que se adaptan mejor a las características de este proyecto (Escalado y coste).

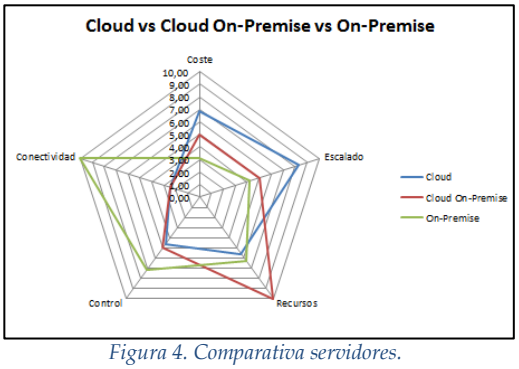

Figura 4. Comparativa servidores.

Los servidores se diseñaron a partir de los requisitos mínimos recomendados oficialmente por las tecnologías desplegadas. La Tabla 3 muestra las prestaciones de los servidores creados.

Implementa	SO	Discos Duros	RAM	CPU
Apache Hadoop	CentOS 7	2x50GB	32GB	8x 2.4 GHz Intel Xeon®
Splunk /ELK	CentOS 7	2x50GB	16GB	4x 2.4 GHz Intel Xeon®

Tabla 3.Software y hardware de los servidores.

Fase 3 – Levantar los servidores: en esta fase se crearon los cinco servidores necesarios para desplegar las tres arquitecturas con éxito. Cuatro servidores para el clúster Hadoop y el servidor restante para Splunk y ELK. Se prepararon los discos duros y se realizó la configuración de las conexiones de red para que los servidores tuvieran visibilidad entre ellos y poder realizar una conexión segura SSH (SecureSHell) desde el ordenador local.

Fase 4 - Desplegar las arquitecturas: en esta fase se descargó, instaló y configuró Cloudera Manager (y sus dependencias funcionales) para tener un clúster Hadoop funcional. Se realizó el mismo proceso con el stack ELK y con Splunk, de modo que las tres arquitecturas y sus módulos pudieran funcionar. Al concluir esta fase ELK y Splunk se podían comunicar con el clúster Hadoop.

Fase 5 - Testear funcionamiento de las arquitecturas: en esta fase se comprobaron todas las funcionalidades de las tres arquitecturas. En esta fase se observó que todas las acciones de Splunk se realizan usando su interfaz web, a diferencia de ELK que requiere ejecutar constantemente comandos en el terminal del servidor.

Para extraer los campos de un evento con Logstash, se utilizaron expresiones regulares, *plugins* y códecs manualmente. Splunk realizó la extracción con un método automático y visual.

Fase 6 - Adaptar las arquitecturas al caso de uso: en esta fase se configuraron las arquitecturas para implementar el caso de uso. Creando las expresiones regulares para extraer los campos de los eventos de sistema en ambas arquitecturas.

Fase 7 - Análisis de los datos y creación de *Dashboards:* se indexaron todos los *logs* proporcionados por el caso de uso para analizarlos mediante ambas arquitecturas y crear *Dashboards* con las visualizaciones pertinentes. Una vez indexados todos los datos, se procedió a realizar un análisis a partir de los requisitos del caso de uso. Las siguientes visualizaciones se añadieron a un *dashboard* tanto en Kibana como Splunk:

- **Comparativa de la cantidad de errores por servidor.** La visualización de los errores que ocurren en cada servidor es un factor clave para optimizar el rendimiento de los mismos
- **Usuarios generadores de más errores por servidor (TOP 10).** La visualización de la cantidad de errores que genera cada usuario, permite investigar el origen de estos errores y poder así reducirlos.
- **Porcentaje de incumplimiento de los SLA asociado a las funciones de Login y PopUp.** Determinar el nivel de cumplimiento de los SLA es un factor determinante para el éxito de una empresa, ya que el incumplimiento puede provocar grandes compensaciones económicas, se pueden incluso rescindir contratos por culpa de esta tasa.
- **Carga por servidor, en términos del *elapsed time* gestionado.** Esto es útil para analizar el tiempo total de procesamiento de las transacciones.
- **Carga por servidor, en términos de eventos totales gestionados.** Permite analizar el volumen de eventos gestionados por servidor.

Determinar la carga de los servidores mediante las dos métricas mencionadas, *elapsed time* y eventos totales, permite determinar si el balanceador de carga funciona correctamente. Los dashboards son herramientas clave para tener una visión global de un sistema o empresa, en este caso sirven para visualizar el rendimiento de los servidores generadores de logs, así como visualizar como el balanceador de carga gestiona la carga de trabajo entre ambos servidores. En la Figura 1 muestra el dashboard generado en Splunk. La Figura 2 muestra el dashboard generado en Kibana.

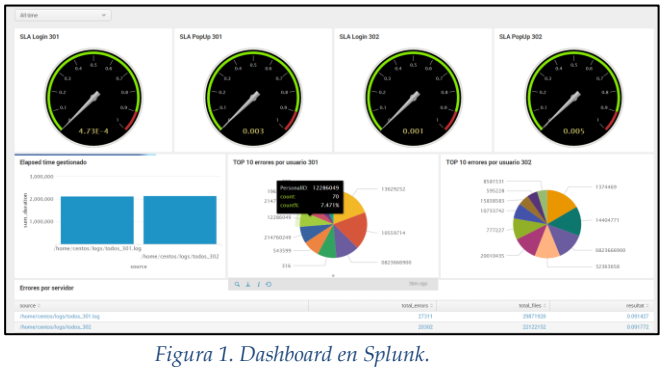

Figura 1. Dashboard en Splunk.

Figura 2. Dasboard en Kibana.

Fase 8 - Análisis de las arquitecturas: se realizó un análisis del rendimiento obtenido al desplegar el *stack* ELK y Splunk a partir de las métricas mostradas en la Tabla 4.

Métricas objetivas	Métricas subjetivas
Velocidad de indexación (Eventos por segundo)	Dificultad para desplegarlas
Uso de CPU	Conocimiento necesario para usarlas
Uso de memoria	Adaptación al caso de uso

Tabla 4. Métricas de análisis.

6 Resultados y análisis

Una vez completados todos los objetivos establecidos se pueden responder a las preguntas planteadas en el proyecto.

¿Cuál de las dos arquitecturas obtiene un mejor rendimiento?

Para reducir el coste total del proyecto, se analizaron las distintas modalidades de servidores existentes, y se decidió desplegar las arquitecturas usando servidores Cloud tradicionales usando los servicios de Amazon Web Services (AWS). La elección del hardware cumplía las especificaciones recomendadas tanto por ELK cómo por Splunk, no obstante, al desplegar las arquitecturas y ponerlas a prueba a partir del caso de uso, se observó que el stack ELK presentaba un rendimiento irregular al indexar los eventos, tal como se muestra en la Figura 3.

En el mejor de los casos ELK ha indexado 3100 EPS, en el peor de los casos 20 EPS, en comparación, Splunk ha indexado a una velocidad constante de 25000 EPS aproximadamente.

Tras realizar una investigación exhaustiva y consultar foros tanto de la comunidad de ELK como la de AWS, se ha llegado a la conclusión que el problema reside en utilizar servidores Cloud tradicionales. No se dispone del hardware físicamente, se trata de hardware virtualizado, con las penalizaciones de rendimiento que ello conlleva.

En la documentación oficial de Elasticsearch no aparece mención alguna a los problemas de rendimiento que esto pueda causar, pero aparecen menciones en los foros de la comunidad de usuarios.

Splunk obtiene un rendimiento elevado y constante, tal como se puede observar en la Figura 4. En su documentación oficial si se hace mención a los problemas de rendimiento que pueden tener lugar al utilizar virtualización de hardware[10].

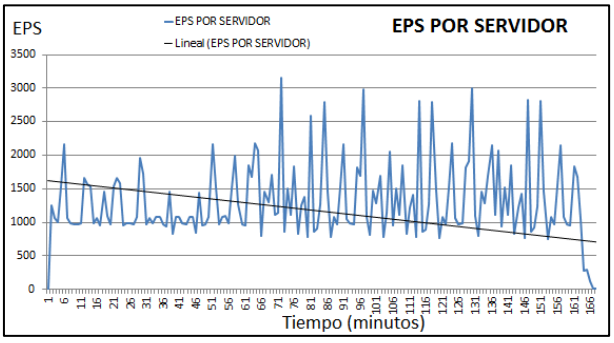

Figura 3.Eventos por segundo ELK.

En un entorno de producción no se pueden producir estas variaciones en cuanto los EPS (Eventos Por Segundo) indexados, ya que resultaría muy complicado dimensionar los servidores para adaptarse a las necesidades del caso de uso. Para obtener un rendimiento más constante, se podrían usar servidores *on-premise* o Cloud *on-premise*. Ya que tal como se analizó, dichos servidores proveen de un mejor rendimiento.

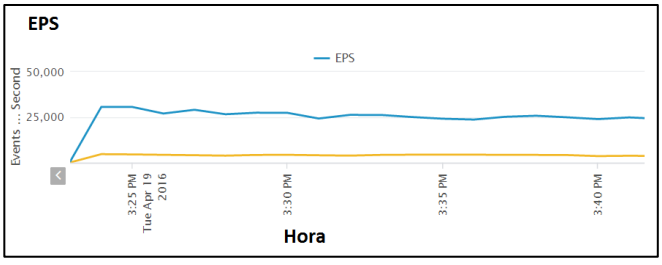

Figura 4. Eventos por segundo Splunk.

La virtualización de hardware penaliza el rendimiento global en un 30% aproximadamente. Además, el uso en entornos Cloud (como en este caso) penaliza el rendimiento de la indexación en un 50% y un 15-20% en velocidad de las consultas.

¿Cuáles son las limitaciones de cada una de las arquitecturas?

Splunk no presenta ninguna limitación. Todas las limitaciones encontradas son del *stack* ELK, tal como muestra la Tabla 5 .

	Limitación
ELK	El usuario mediante *testing* tiene que comprobar si se realiza correctamente la indexación.
ELK	El usuario no puede identificar los eventos que no han sido indexados.
ELK	En servidores Cloud tradicionales no permite indexar los eventos en un tiempo razonable.
ELK	La interacción con Hadoop no es nativa ni bilateral.
ELK	No dispone de HTTPS o controles de acceso basado en roles y agentes.
ELK	No permite crear alertas.
ELK	No permite crear reportes automáticos cada cierto tiempo.
ELK	No permite visualizar en tiempo real directorios en HDFS.
ELK	Tratar los eventos de múltiples líneas es complicado y afecta al rendimiento.
ELK	Para indexar datos de HDFS a Elasticsearch, estos tienen que estar estructurados. . Por lo tanto se tiene que usar un proceso bash para estructurar los datos, tal que la propiedad de near-real time usando el conector es-hadoop no es factible (no se lleva a cabo).
ELK	Al agotar el espacio en disco: el pipeline de Logstash se detiene, Kibana no permite visualizar los eventos y Elasticsearch se vuelve inestable, provocando la corrupción de índices.
Kibana	Sistema monousuario.
Kibana	Imposibilidad de post-procesar los eventos para calcular valores a partir de ellos o bien crear nuevos campos.
Logstash	El proceso de extracción de campos exige utilizar filtros que no son *thread-safe*. El rendimiento sufre una penalización al no paralelizar el proceso mediante *multithreading*.
ELK	Imposibilidad de realizar auditorías mediante el uso de herramientas de gestión.

Tabla 5. Limitaciones tecnológicas.

¿Cuál sería el escenario óptimo para usar cada arquitectura?

La decisión de que arquitectura usar no es trivial, hay que tener en cuenta muchos factores (volumetría de datos, presupuesto…).

El stack ELK puede ser una buena opción para casos de uso sencillos, que no requieran de funcionalidades avanzadas ni tratar datos confidenciales, implementar controles de acceso… Puede ser desplegado en PIMES que quieran implementar Big Data sin tener que dedicar grandes recursos económicos a ello.

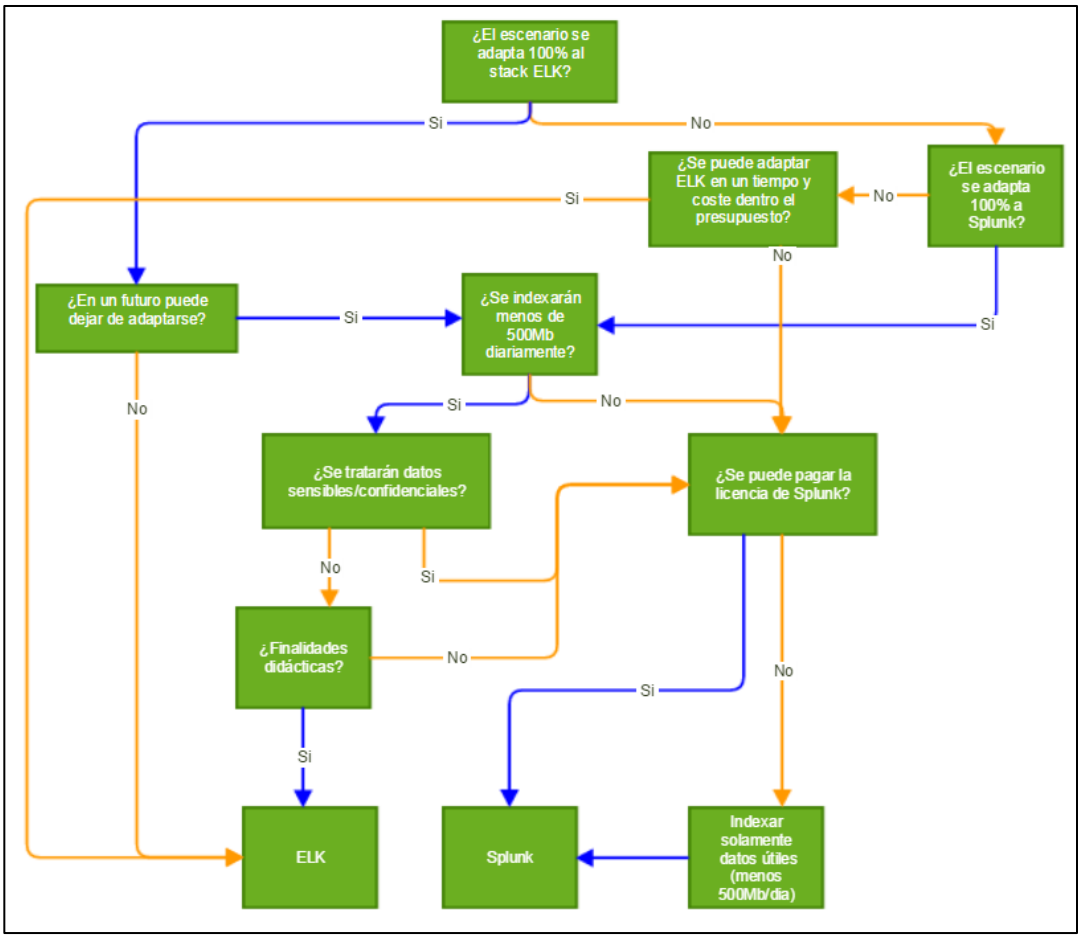

Figura 5.Diagrama de decisión ELK y Splunk.

Splunk es una herramienta muy versátil, aunque no es de código abierto, permite adaptarla a las necesidades de cualquier caso de uso siendo una solución muy madura. Splunk prioriza el rendimiento, la capacidad de ser real time, la privacidad y seguridad mediante controles de acceso mediante el uso de roles y agentes. En la Figura 5 se contemplan distintos parámetros para ayudar a tomar la decisión.

¿Dado un caso de uso concreto, qué arquitectura se adaptaría mejor y con más facilidad?

El caso de uso implementado, requiere de funcionalidades avanzadas, y por lo tanto ELK no se ha adaptado correctamente. Los logs proporcionados siguen un patrón no estándar, no están en formato .csv, en la Figura 6 se representa el único flujo de datos posible y que permite la interacción entre el Clúster Hadoop y el servidor de ELK.

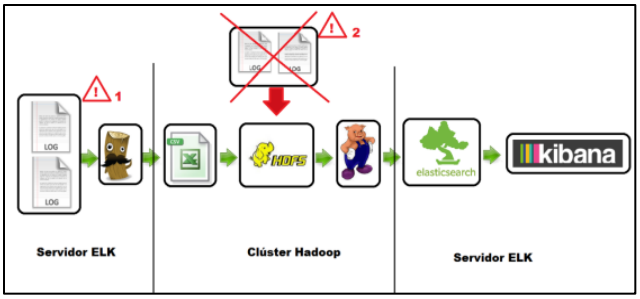

Figura 6.Flujo de datos entre ELK y Hadoop.

Las advertencias de la Figura 6 hacen referencia al incumplimiento de las siguientes especificaciones del caso de uso:

1. Los ficheros de logs tenían que estar almacenados en HDFS conservando su formato original, sin ser modificados ni duplicados en el servidor de ELK.

2. La conectividad entre ELK y HDFS requería que los logs estuvieran estructurados.

Desde que el conector no interactúa con Logstash, se concluye que aunque los ficheros de logs estén estructurados es imposible aprovechar el stack ELK. El conector no es una solución madura, para que lo fuera, se tendría que adaptar a Logstash, permitiendo añadir como input de Logstash un directorio en HDFS.

Dadas las limitaciones de rendimiento que presenta el stack ELK desplegado en AWS, se ha testeado el flujo de datos representado en la Figura 6, aun incumpliendo las especificaciones del caso de uso. Es un flujo funcional, pero no es real time, ya que se necesitan procesos bash para automatizar la conversión de los logs en .csv y para ejecutar las sentencias con Pig para indexar los eventos en Elasticsearch.

En cuanto a Splunk, este se ha adaptado perfectamente al caso de uso, permitiendo indexar toda la serie histórica de datos. Además, dada la volumetría de datos que generan los servidores analizados, no se requiere adquirir una licencia para seguir monitorizando los logs de sistema. La conectividad entre Splunk y Hadoop es out-of-the-box.

La Tabla 6 muestra las características más destacables de cada arquitectura.

Característica		
Splunk	Configurar la herramienta para tratar los eventos de múltiples líneas es trivial.	✓
Splunk	Desde que se configura el *input* se pueden consultar los datos.	✓
Splunk	El campo *elapsed time* se calcula en post-procesado, sin perjudicar la indexación.	✓
Splunk	La interacción con Hadoop es nativa y bilateral.	✓
Splunk	Permite comprobar la extracción de campos antes de realizar la indexación.	✓
Splunk	Permite crear alertas en función de los eventos o campos, y enviar emails automáticamente.	✓
Splunk	Permite crear reportes automáticos cada cierto tiempo.	✓
Splunk	Permite explorar el *input* antes de indexar y extraer los campos.	✓
Splunk	Permite guardar modelos de datos para agilizar las futuras consultas.	✓
Splunk	Permite indexar todos los eventos sin perdidas.	✓
Splunk	Permite usar HTTPS, ocultación de datos y controles de acceso.	✓
Splunk	Permite visualizar directorios en HDFS en tiempo real.	✓
Splunk	Permite visualizar los datos en tiempo real.	✓
Splunk	Velocidad de indexación óptima, en horas puede indexar eventos generados durante años.	✓
ELK	El campo *elapsed time* se calcula durante la extracción de campos (afecta al rendimiento).	✗
ELK	En servidores Cloud tradicionales no permite indexar los eventos en un tiempo razonable.	✗
ELK	No permite explorar el input antes de indexar y extraer los campos.	✗
ELK	No permite guardar modelos de datos para agilizar las consultas.	✗
ELK	No permite visualizar directorios en HDFS en tiempo real.	✗
ELK	Visualización de datos *near-real* time.	✗
ELK	Tratar los eventos de múltiples líneas es complicado y afecta al rendimiento.	✗

Tabla 6. Características destacables de cada arquitectura.

¿Es más rápido desplegar una arquitectura open-source o una de pago?

Ha sido más rápido desplegar la arquitectura de pago que la open-source. Ya que ELK requería ser modificado para adaptarse al caso de uso. La herramienta de pago contemplaba todas las funcionalidades requeridas.
En software es muy común esta situación. Si se selecciona un sistema operativo de pago (Windows) y uno open-source (CentOS), y se analiza la cuota de mercado de cada uno, la herramienta de pago es más popular. Esta está adaptada a un público más amplio, tiene mejor financiación, más personal trabajando, y su usabilidad es más sencilla. No quiere decir que la gente no use software open-source, pero no es accesible por todos, se requieren conocimientos específicos y avanzados para poder ser adaptado a las necesidades de cada persona o empresa. Cuando el coste del software de pago es asequible y cumple las expectativas y requerimientos, se recurre a él, antes que el software open-source.

Resultados de la planificación

La planificación realizada del proyecto se ha llevado a cabo en el tiempo establecido. Se cuantificó adecuadamente la duración de las fases, así como las tareas asociadas a cada una de ellas.

Debido a que en la planificación del proyecto se añadieron holguras, y la carga inicial de 5GB de ficheros de logs fue satisfactoria en ambas arquitecturas, se cargaron 5GB extras en logs. Esta decisión de ingestar más datos sirvió para analizar los problemas de rendimiento observados en el *stack* ELK.

7 Conclusión

A partir del desarrollo del proyecto se concluye que en algunos escenarios las soluciones open-source son mejores que las soluciones de pago, no obstante, existe una realidad inherente. Las soluciones de pago gozan de una mejor financiación, y están diseñadas para satisfacer a un amplio público. Las soluciones open-source tienen una filosofía de best-efford, ya que los colaboradores participan en el proyecto como un acto altruista, sin beneficio económico y en su tiempo libre. Esta diferencia afecta la calidad y complejidad del software desarrollado.

Al poner a prueba el stack ELK y Splunk ha quedado patente esta diferencia. Splunk es una solución robusta, madura y adaptable a cualquier caso de uso, con un rendimiento óptimo bajo cualquier infraestructura de servidores. Ha sido diseñado para que la interacción entre el usuario y el sistema sea fácil e intuitiva, de modo que no se pierda tiempo levantando y configurando los parámetros necesarios para que todo funcione. Aún con esta facilidad de interacción se pueden generar consultas realmente complicadas, gracias al lenguaje SPL.

El stack ELK a nivel teórico puede llegar a adaptarse a cualquier caso de uso, al poder modificar el código fuente. Pero, hay que tener presente el coste derivado de adaptar el sistema, ya que puede superar los costes de una licencia de Splunk con facilidad. Para poder implementarse en un entorno real, Elasticsearch se tiene que adaptar a las necesidades de la empresa, y por defecto no las cubre. Para poderse adaptar, se precisará de un gran equipo con grandes conocimientos en Java, tiempo y motivación. ELK se adapta mejor a una pequeña empresa, con casos de uso simples, que no precisen controles de acceso, con logs de sistema que no requieran de grandes expresiones regulares para extraer los campos ni el uso de plugins para calcular campos nuevos, ya que tal como se ha observado, se sufre una gran penalización en el ren-

dimiento.

El motivo de peso determinante para usar ELK en lugar de Splunk puede ser el presupuesto de la empresa. Si la empresa desea indexar más de 500Mb de datos diariamente y no puede pagar la licencia de Splunk, debe usar ELK (siempre y cuando las funcionalidades sean out-of-the-box). Si desean indexar menos cantidad de datos, pueden usar la licencia gratuita que ofrece Splunk.

El caso de uso implementado proveía de un conjunto de datos cercanos a los 10Gb. Este conjunto de datos ha sido generado en dos años, por lo que diariamente se generaban aproximadamente 14Mb de datos. Se han podido indexar todos los datos en la licencia de prueba, y se podría usar la licencia gratuita de Splunk (límite de 500Mb diarios) para analizar los datos en tiempo real una vez finalizada la licencia de prueba.

En un futuro puede ser que ELK evolucione y se sitúe líder en el mercado, pero actualmente no lo es. Y la mejor solución es usar Splunk, ya que tal como muestra la Figura 7, Splunk tiene mayor cobertura en todas las dimensiones analizadas.

En el caso que no se pueda pagar una licencia de Splunk, existe la opción de ingestar sólo los datos útiles, evitando indexar datos y campos que no sirvan para obtener conocimiento.

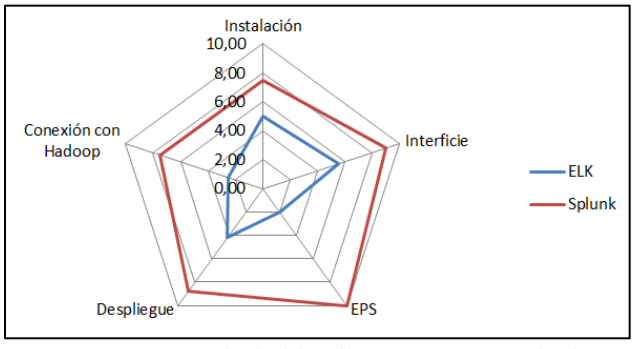

Figura 7. Resultado del análisis entre ELK y Splunk.

8 Conclusiones personales

Trabajar en un proyecto aplicado es una experiencia muy gratificante y exigente, ya que hay relativamente poco tiempo por la envergadura del proyecto. Normalmente en este tipo de proyectos, hay un equipo de trabajo de tamaño considerable, en este caso yo he desarrollado la totalidad del proyecto.

Encontré grandes dificultades al desplegar el *stack* ELK, en especial al realizar la extracción de campos de los eventos, al realizar el cálculo de tiempo transcurrido entre funciones lanzadas por el mismo usuario, y también al unir eventos de varias líneas en uno solo.

Los estudios cursados en ingeniería informática han sido vitales para afrontar los problemas citados y muchos más, he aplicado conocimientos de prácticamente todas las asignaturas cursadas en este grado: configuración de redes, aritmética modular, programación, gestión de sistemas operativos, gestión de proyectos…

Realizar un trabajo de esta envergadura te hace crecer personalmente, ya que acabas superando todos los retos que te planteas.

Si la realización de un trabajo de final de grado no fuera parte del plan de estudios, lo realizaría de todas formas, y recomendaría a otras personas hacer lo mismo, porqué aprendes y consolidas los conocimientos adquiridos durante el período académico a la vez que desarrollas nuevos hábitos.

9 Agradecimientos

Agradecer el soporte constante de la tutora de este proyecto, Katerine Diaz, su ayuda y consejos han sido vitales para el correcto desarrollo de este trabajo. Mención especial para Everis España, empresa que ha asumido todos los gastos del proyecto, así como a la entidad privada que ha proporcionado el caso de uso y los *logs* de sus sistemas. También mencionar a Marc Sangüesa, líder del centro de excelencia en Big Data, donde he efectuado el desarrollo del proyecto, por sus consejos e incondicional soporte.

10 Bibliografía

[1] Web – Amount of connected devices [Consulta 01/06/16] Disponible en: http://www.globalwebindex.net/blog/digital-consumers-own-3.64-connected-devices

[2] Web – Forbes data analytics [Consulta 02/06/16] Disponible en: http://www.forbes.com/sites/louiscolumbus/2015/03/15/data-analytics-dominates-enterprises-spending-plans-for-2015/#482215db3eb4

[3] Web – Cost to deploy Big Data [Consulta 02/06/16] Disponible en: http://www.sas.com/resources/asset/Big-Data-in-Big-Companies.pdf

[4] Blog – Popularity of Cloudera and Hadoop [Consulta 03-06/16] Disponible en: https://vision.cloudera.com/the-rise-of-the-apache-hadoop-data-warehouse-gartner-magic-quadrant-2016/

[5] Web – Sistema HDFS [Consulta 06-06/16] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[6] Web – Stack ELK https://www.elastic.co [Consulta 13-06/16]

[7] Web – Splunk http://www.splunk.com/es_es [Consulta 05-05/16]

[8] Paper – NASA Work Breakdown Structure [Consulta 23-04/16] Disponible en: https://www.nasa.gov/sites/default/files/files/CEH_AppB.pdf

[9] Web - Productos de Amazon Web Services. [Consulta 26/02/2016] Disponible en: https://aws.amazon.com/es/products/

[10] Web - Penalización rendimiento hardware Splunk [Consulta 04/05/2016] Disponible en: http://docs.splunk.com/Documentation/Splunk/6.0/Installation/Referencehardware

[11] Web – Search Processing Language [Consulta 01/05/2016] Disponible en: https://docs.splunk.com/Splexicon:Searchprocessinglanguage

[12] Web – Daily amount of data [Consulta 02/06/16] Disponible en: http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[13] Artículo – Ritmo de crecimiento de los datos [Consulta 02/06/16] Disponible en: http://www.computerworld.es/tendencias/la-cantidad-de-datos-que-se-genera-en-el-mundo-se-duplica-cada-dos-anos