

**T.F.M.**  
**Procesamiento de Datos en Tiempo Real.**



Araceli Macía Barrado  
Abril 2017

CIFF Business School  
Master en Big Data y Business Inteligent  
Edición 2015 / 2016

### **Agradecimientos**

Quiero dar las gracias a mi marido, ya que gracias a su apoyo y ayuda pude tomar la decisión de cambiar de rumbo profesional y poder realizar el Máster que he querido.

También quiero agradecer a Alfonso Campos y a Strategy Big Data la oportunidad de poder realizar este proyecto, la ayuda, tutela, así como su generosidad al haberme abierto las puertas a sus oficinas para poder desarrollarlo.

## Abstract

El avance de las tecnologías móviles y conexiones a Internet hace que el número de usuario en redes sociales esté continuamente creciendo, lo que permite que cualquier usuario en cualquier sitio pueda publicar y compartir una foto, comentario o video manifestando un hecho en una red social que llega a ser un escaparate para conocimiento a nivel mundial. Es por ello que las empresas hoy en día tengan que prestar cuidadosa atención y despertar al hecho de que ya no valen los modelos de negocio tradicionales, sino que deben salir de su zona de confort y explorar nuevas formas de conectar y acercarse a la sociedad para estar al tanto de los comentarios acerca de sus productos y servicios y poder atajar problemas o fomentar situaciones que favorezcan sus intereses. Ante esta necesidad de negocio de conocer los datos según se están produciendo, surgen y evolucionan tecnologías que permiten obtenerlos y procesarlos en tiempo real.

Esto es lo que se va a desarrollar en este proyecto, se evaluarán redes sociales para ver cuál es la que mejores datos puede aportar a una empresa, se presentarán los factores a tener en cuenta, así como también se presentarán distintas tecnologías que permiten realizar procesamiento en tiempo real, para al final presentar un diseño de una arquitectura con las herramientas tecnológicas adecuadas que permitan cumplir con estos factores y requisitos para obtener la información necesaria en el momento adecuado que pueda hacer que una empresa pueda atajar un mal comentario o conocer cuáles son los influencers que pueden ayudarle.

*Palabras clave:* Redes Sociales, Empresas, Nuevas formas de conectar, Procesar en tiempo real

## Tabla de Contenidos

<b>1. Introducción y objetivo.....</b>	<b>8</b>
<b>1.1. Introducción .....</b>	<b>8</b>
<b>1.2. Objetivo.....</b>	<b>11</b>
<b>2. Estado del arte.....</b>	<b>12</b>
<b>2.1. Redes Sociales.....</b>	<b>12</b>
2.1.1. Facebook .....	14
2.1.2. Twitter.....	15
2.1.3. Instagram.....	16
<b>2.2. Procesamiento en tiempo real y Streaming.....</b>	<b>16</b>
2.2.1. Arquitectura Streaming.....	17
2.2.2. Tecnologías existentes .....	21
<b>3. Análisis de alternativas.....</b>	<b>33</b>
<b>3.1. Red Social .....</b>	<b>33</b>
3.1.1. Diferencias entre Facebook, Instagram y Twitter.....	33
<b>3.2. Análisis de las tecnologías .....</b>	<b>35</b>
3.2.1. Adquisición de datos.....	35
3.2.2. Procesamiento de datos.....	37
3.2.3. Capa de almacenamiento .....	40
<b>4. Diseño .....</b>	<b>41</b>
<b>4.1. Selección de Red Social.....</b>	<b>41</b>
<b>4.2. Tipos de Tuits .....</b>	<b>44</b>
<b>4.3. Factores a tener en cuenta.....</b>	<b>56</b>
<b>4.4. Diseño técnico.....</b>	<b>58</b>
<b>4.5. Solución técnica.....</b>	<b>62</b>
4.5.1. Diagrama de alto nivel .....	62
4.5.2. Código implementado .....	62
<b>5. Evaluación .....</b>	<b>73</b>
<b>5.1. Inserciones y modificaciones.....</b>	<b>75</b>
<b>5.2. Evaluar tiempos de procesamiento.....</b>	<b>76</b>
<b>5.3. Resumen de la evaluación .....</b>	<b>82</b>
<b>6. Conclusiones .....</b>	<b>83</b>
<b>7. Referencias.....</b>	<b>87</b>

### **Lista de tablas**

Tabla 1. <b>Objeto Tweet.</b> (Twitter, Tweets, s.f.) .....	52
Tabla 2. <b>Objeto Place</b> (Twitter, Places, s.f.) .....	53
Tabla 3. <b>Objeto usuario.</b> (Twitter, Users, s.f.) .....	53
Tabla 4. Resumen de evaluación .....	82

## Lista de figuras

Figura 1. Estadísticas de uso de redes sociales. (Brandwatch, 2016) .....	9
Figura 2. Estadística de negocios y redes sociales. (Brandwatch, 2016) .....	10
Figura 3. Previsión usuarios España (statista, s.f.) .....	12
Figura 4. Uso de redes sociales en España (We Are Social, 2016) .....	13
Figura 5. Redes utilizadas para seguir marcas .....	14
Figura 6. Arquitectura Streaming .....	18
Figura 7. Conceptos Kafka (Perea, Captura y Almacenamiento Intermedio) .....	22
Figura 8. Componentes Flume .....	23
Figura 9. Componentes Apache Storm (IBM, s.f.) .....	25
Figura 10. Componentes Flink (Gamboa, Introducción a Apache Flink, s.f.) .....	27
Figura 11. Componentes Spark (Gamboa, Introducción a Apache Spark – Batch y Streaming, s.f.) .....	28
Figura 12. Componentes Cassandra .....	31
Figura 13. Componentes MongoDB .....	32
Figura 14. GitHub Apache Storm y Apache Flink .....	38
Figura 15. Comparativa Storm y Spark (Vukelic, 2014) .....	38
Figura 16. GitHub Apache Storm y Apache Spark .....	39
Figura 17. GitHub Apache Flink y Apache Spark .....	40
Figura 18. Ejemplo de Timeline (Eje Central, 2016) .....	44
Figura 19 Creación de un tuit .....	44
Figura 20. Tuit convencional .....	45
Figura 21. Tuit Summary card .....	45
Figura 22. Tuit Sumary Card con Imagen .....	46
Figura 23. Tuit App Card .....	46
Figura 24. Tuit Player Card .....	47
Figura 25 Trending Topic 19.03.2017 (Trendinalia, s.f.) .....	49
Figura 26 Ejemplo histograma hashtag día ( <a href="http://www.tweet-tag.com/">http://www.tweet-tag.com/</a> , s.f.) .....	49
Figura 27. Ratio de entrada hashta dia .....	50
Figura 28 Ejemplo histograma hashtag programa nocturno .....	50
Figura 29. Ratio de entrada hashtag programa nocturno .....	51
Figura 30. Ratio entrada tuits .....	51
Figura 31. Datos de Tuit, momento 0 .....	54
Figura 32. Datos de Tuit, momento 1 .....	55
Figura 33. Datos de Tuit, momento 2 .....	55
Figura 34. Sources Data Streaming (Spark, s.f.) .....	60
Figura 35. Diagrama alto nivel solución .....	62
Figura 36. Adquisición Datos. Código 1 .....	64
Figura 37. Adquisición Datos. Código 2 .....	64
Figura 38 Adquisición Datos. Código 3 .....	65
Figura 39. Adquisición Datos. Código 4 .....	65
Figura 40. Procesamiento, código 1 .....	66
Figura 41. Procesamiento, código 2 .....	66

Figura 42. Procesamiento, código 3 .....	66
Figura 43. Procesamiento, código 4 .....	67
Figura 44. Procesamiento, código 5 .....	67
Figura 45. Procesamiento, código 6 .....	67
Figura 46. Procesamiento, código 7 .....	68
Figura 47. Procesamiento, código 8 .....	68
Figura 48. Procesamiento, código 9 .....	68
Figura 49. Procesamiento, código 10 .....	69
Figura 50. Procesamiento, código 11 .....	69
Figura 51. Almacenamiento, código 1 .....	70
Figura 52. Almacenamiento, código 2 .....	70
Figura 53. Almacenamiento, código 3 .....	70
Figura 54. Almacenamiento, código 4 .....	71
Figura 55. Almacenamiento, código 6 .....	72
Figura 56. Almacenamiento, código 7 .....	72
Figura 57. Almacenamiento, código 8 .....	72
Figura 58. Almacenamiento, código 9 .....	73
Figura 59. Evaluación inserción 1 .....	75
Figura 60. Evaluación inserción 2 .....	75
Figura 61. Evaluación inserción 3 .....	75
Figura 62. Evaluación inserción 4 .....	75
Figura 63. Evaluación inserción 5 .....	76
Figura 64. Evaluación inserción 6 .....	76
Figura 65. Evaluación tiempos 1 .....	77
Figura 66. Evaluación tiempos 2 .....	77
Figura 67. Evaluación tiempos 3 .....	78
Figura 68. Evaluación tiempos 4 .....	78
Figura 69. Evaluación tiempos 5 .....	79
Figura 70. Destino de datos, código 1 .....	80
Figura 71. Destino de datos, código 2 .....	80
Figura 72. Destino de datos, código 3 .....	81
Figura 73. Destino de datos, código 4 .....	81
Figura 74. Destino de datos, código 5 .....	81
Figura 75. Cuadro Comparativo Tecnologías Streaming .....	84

## 1. Introducción y objetivo

### 1.1. Introducción

El caso de negocio al que busca dar solución este proyecto, nace de realizar una reflexión sobre el mundo o sobre el entorno en que nos movemos, cuyas necesidades van cambiando a un ritmo casi frenético.

Por un lado, vivimos en un mercado global, lo que en economía significa que se caracteriza por la integración de las economías locales en una economía de mercado mundial (Wikipedia, 2003) sin fronteras, donde un inversionista de un país puede escoger invertir en una empresa de otro país, y una empresa puede utilizar recursos más baratos en otros países para ser más competitiva en cuanto a costes. (Dotras, 2014)

Como consecuencia de esto, las empresas tienen más competencia, dado que no solo compiten contra otras empresas en el ámbito nacional, sino que compiten con otras empresas a nivel mundial. Por lo que, para sobrevivir, se está demostrando que ya no valen las estructuras empresariales que tenían antes, sino que las empresas que están sobreviviendo son las más innovadoras, que mejor se adaptan y se apoyan en nuevas tecnologías. (Pascual & Ropero, 2015)

El tema es que hasta hace relativamente poco, los datos y las tecnologías existentes ofrecían la información una vez que se había producido, y pasaba un tiempo entre que la información se producía y entre que la información se utilizaba para la toma de decisiones. Con lo que cuando se tomaba una decisión que afectaba al futuro de la empresa, pudiera darse el caso, de que el dato ya no era real o estuviera caducado, y el movimiento o la decisión a realizar ya no fuera el adecuado, o ya fuera tarde para poder mejorar nada.

Unido a esta realidad, tenemos otras, ya que el mundo avanza en muchos sentidos, pero centrándonos en dos, que son en los que se basa la reflexión de este proyecto:

- 1) El uso de las redes sociales está creciendo en la sociedad a nivel mundial a un ritmo exponencial, ya que las aplicaciones sociales se presentan de forma muy intuitiva y accesible para la sociedad, sobre todo gracias a los avances en lo que a móviles se refiere, y son además un medio fácil para obtener información, por lo cual cada vez más público las utiliza y las prefiere como herramienta de comunicación. Esto hecho se puede ver en la siguiente figura se puede ver el crecimiento de uso en las redes sociales en el año 2016, extracto del estudio realizado por la empresa Brandwatch, donde se destaca que hay 12 usuarios cada segundo.

## Estadísticas de redes sociales

- Para entrar en contexto, en marzo de 2016 la población mundial era de [7,4 mil millones](#)
- Internet tiene [3,17 mil millones de usuarios](#)
- Hay [2,3 mil millones de usuarios activos](#) en redes sociales
- El [91% de las marcas de retail](#) usan dos o más canales de redes sociales
- Los usuarios de internet tienen [5,54 cuentas en redes sociales](#) de promedio
- Los usuarios de redes sociales crecieron [176 millones](#) el año pasado
- Hay 1 millón de usuarios activos de redes sociales en móviles nuevos cada día. Es decir, [12 cada segundo](#)
- Facebook Messenger y Whatsapp manejan [60 mil millones de mensajes diarios](#)

*Figura 1. Estadísticas de uso de redes sociales. (Brandwatch, 2016)*

Como se puede ver en la siguiente figura, las redes se presentan como un canal más de venta del que hacen uso las empresas para publicitar sus productos y servicios.

## Estadísticas de negocios y redes sociales

- Las redes sociales obtuvieron ingresos por publicidad de [8,3 mil millones de dólares](#) en 2015
- El 38% de las organizaciones destinaron un [20% de su presupuesto total de publicidad](#) a canales de redes sociales
- Solamente 20 empresas de Fortune 500 interactúan con sus clientes en Facebook, mientras que el [83% tiene presencia en Twitter](#)
- El [96% de las personas](#) que hablan sobre una marca en redes sociales no siguen el perfil de esa marca
- Un 78% de las personas que se quejan de alguna marca en Twitter esperan una respuesta [en una hora o menos](#)

*Figura 2. Estadística de negocios y redes sociales. (Brandwatch, 2016)*

Sin embargo, la clave para las empresas está en acceder a las redes sociales con el fin de extraer y procesar información de lo que los usuarios comentan, ya que un mal comentario acerca de un producto o servicio, puede afectar negativamente la imagen de una compañía. Un ejemplo de esto se puede encontrar en lo que ocurrido recientemente con la compañía United Airlines en cuanto a la fuerza bruta empleada para sacar a un pasajero del avión. La crisis de reputación que está viviendo ahora la compañía no habría sido la misma si no fuera porque la gente lleva móvil y acceso a Internet y por tanto acceso a redes sociales lo que permitió denunciar y mostrar lo ocurrido a todo el mundo. (Olmo, 2017). Que la empresa no haya sabido reaccionar a tiempo y atajar este tema, le ha causado por el momento pérdidas multimillonarias.

### 2) Avances tecnológicos:

- Aparecen y evolucionan tecnologías Big Data que permiten procesar y analizar cantidades ingentes de información.
- Productos que permiten analizar y visualizar la información que generan las propias aplicaciones informáticas de las empresas, sin tener que invertir en desarrollos a medida más costosos.

- Herramientas para obtener información en tiempo real ya sea de logs, eventos...
- Medios de conexión ofrecidos por las propias redes sociales, que permiten acceso de primera mano a las opiniones y gustos del público en general.

Uniendo estos dos aspectos que están estrechamente relacionadas, *una empresa puede tener ventaja competitiva por el hecho de poder obtener información de las redes sociales en tiempo real*, es esto lo que dará el conocimiento para poder tomar las decisiones adecuadas en el momento oportuno, mejorar y optimizar procesos, incrementar beneficios, acercarse a futuros clientes, es decir tendrá más oportunidades de sobrevivir en este mundo de mercado global que se comentaba anteriormente.

Este proyecto, por tanto, se centra en proporcionar una solución que apoyándose en nuevas tecnologías Big Data, explote los datos de la gran fuente de información que representan las redes sociales, y que además lo haga en tiempo real.

## **1.2. Objetivo**

El objetivo del TFM es dar respuesta a la necesidad expuesta en cuanto a la necesidad de la información, y mostrar una solución que permita el acceso a una red social, con el fin de *poder procesar y analizar la información extraída en tiempo real*.

Una vez fijado el objetivo, hay que elaborar el plan de acción que nos lleve a conseguirlo. Dicho plan para por:

- Escoger cual es la red social que más información o datos puede aportar para una compañía.
- Escoger cual es la mejor tecnología para acceder a dicha información

## 2. Estado del arte

Este capítulo se centra en ver qué opciones existen en los dos conceptos claves del TFM y para ello se hace lo siguiente:

- Estudio de las redes sociales existentes para determinar cuál sería la red idónea por la información que pueden aportar a una empresa.
- Explicar qué es procesamiento en tiempo Real y Streaming, que lo caracteriza, y cuáles son las tecnologías existentes que permiten realizarlo.

### 2.1. Redes Sociales

Como se comentaba anteriormente, el número de usuarios de las redes sociales sigue aumentando. A continuación, se muestra una previsión de usuarios de redes sociales en España del 2014 a 2018, realizada por Statista.

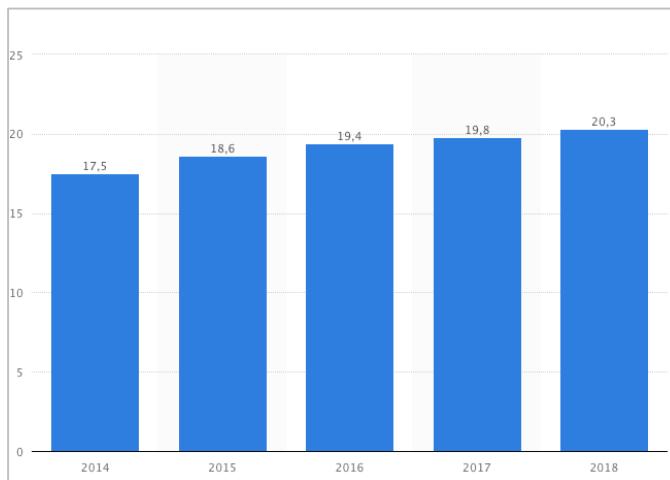


Figura 3. Previsión usuarios España (statista, s.f.)

Como se ve en la figura, el número de usuarios de las redes sociales continúa en aumento.

Se muestra ahora el consumo de usuarios por red social en España, que se ha extraído de We Are Social, que hacen un estudio sobre internet y las redes sociales en el mundo, donde se

puede ver que las redes sociales que calan más en la sociedad es WhatsApp seguido muy de cerca por Facebook.

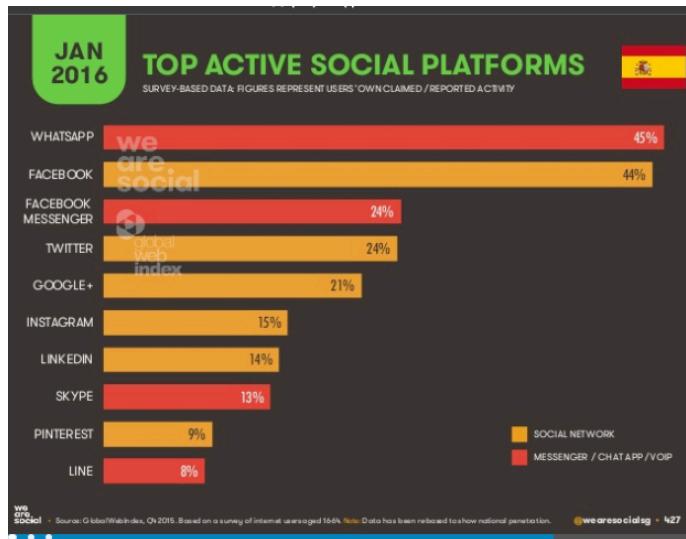


Figura 4. Uso de redes sociales en España (We Are Social, 2016)

El uso de las redes sociales no se limita a usuarios personales, las empresas conocedoras de su potencial, también están haciendo uso de las redes sociales, por ejemplo, generando una página en Facebook o un usuario en twitter, o creando un foro en LinkedIn.

La cuestión es decidir cuál sería la mejor red social de la que extraer la información en tiempo real. Como se ha visto en la figura existen numerosas redes sociales, por lo tanto, para saber escoger cual serían las redes válidas para este proyecto, se hace referencia al estudio realizado por *iab* sobre las redes sociales (IAB\_EstudioRedesSociales\_2016\_VCorta.pdf, 2016), en el que se extrae la siguiente figura:

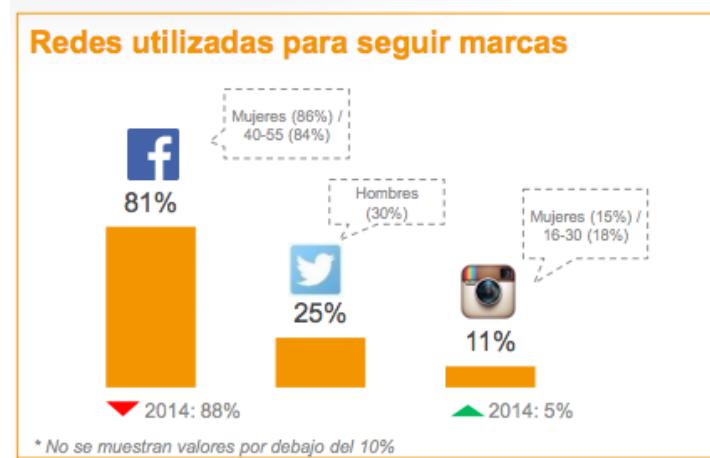


Figura 5. Redes utilizadas para seguir marcas

Por tanto, las redes sociales que se van a considerar en el proyecto son Facebook, Twitter e Instagram, ya que son las redes sociales más utilizadas por parte de los usuarios para seguir marcas y eso las sitúa dentro del objetivo de este proyecto orientado a qué las empresas puedan obtener ventaja competitiva por hacer uso de información de las redes sociales en tiempo real.

### 2.1.1. Facebook<sup>1</sup>

Es la red social más popular, ya que trata simplemente de conectar personas con personas. El espíritu de la red está muy orientado a las vidas personales, donde los usuarios suben fotos, o cuentan cosas de sus estados de ánimo, comentarios personales, gustos, etc. y todo bajo el concepto de la privacidad sobre con quién se comparten las cosas. De forma que el usuario puede decidir quiénes son sus amigos, y dar permisos sobre quién puede ver o no sus fotos, comentarios, etc.

Desde el punto de vista de las empresas, Facebook es una red muy potente por el alto número de usuarios, y porque las páginas de Facebook representan una gran oportunidad para

---

<sup>1</sup> <https://es-es.facebook.com/>

fidelizar a clientes, ya que se trata de un trato muy cercano y personal con los usuarios, y los usuarios pueden decir si les gusta o no les gusta la publicidad que les presentan, con lo cual tendría el feedback en tiempo real. (Rubín, Qué es facebook, cómo funciona y que te puede aportar esta red social, 2016). Por ello las marcas ya se están aprovechando de esto para incorporar anuncios publicitarios en el portal o muro de los usuarios.

### **2.1.2. Twitter<sup>2</sup>**

Twitter es una red social que permite elegir con quien se relaciona un usuario y que limita sus mensajes a 140 caracteres. Sin embargo, a diferencia de Facebook, las relaciones son asimétricas, un usuario puede elegir a quien sigue sin tener que pedir permiso al seguidor. Lo que produce dos listas diferenciadas, la gente a la que sigue un usuario y la gente que sigue ese usuario, que no tienen porque están conectadas. Twitter se puede asemejar a una especie de sala de chat mundial, donde la gente puede comentar libremente y cualquiera puede leerlo, y opinar al respecto. En este sentido, Twitter es una potentísima herramienta comercial para las empresas para ganar reputación, promover productos y servicios (Rubín, Qué es Twitter, cómo funciona y qué te puede aportar esta red social, 2016).

Una de las ventajas de esta red social, es que Twitter pone a disposición de los desarrolladores un API de fácil uso y acceso gratuito, que además está muy bien documentado.

---

<sup>2</sup> <http://twitter.com/>

### **2.1.3. Instagram<sup>3</sup>**

Es una red social donde los usuarios suben fotos o videos, con la opción de aplicar diversos efectos fotográficos para compartirlo con el resto de usuarios. De modo que cualquier usuario puede comentar la foto de otra persona a la que previamente ha marcado para seguir.

En esta red social el uso principal es para hacer seguimiento de celebridades, páginas, empresas y poder ver sus publicaciones. ([www.expertosnegociosonline.com](http://www.expertosnegociosonline.com), s.f.)

## **2.2. Procesamiento en tiempo real y Streaming.**

Una de los primeros puntos a determinar, es definir a que llamamos procesar en tiempo real y que es Streaming, ya que este es el punto fuerte del TFM que se está tratando.

Bajo Tiempo real, se pueden encontrar diferentes definiciones, por ejemplo: (redindustria, s.f.)

- Rápida transmisión y proceso de datos orientados a eventos y transacciones a medida que se producen, en contraposición a almacenarse y retransmitirse o procesarse por lotes.
- Un sistema de tiempo real es aquel capaz de procesar una muestra de señal antes de que ingrese al sistema la siguiente muestra.
- Un sistema de tiempo real es aquel en el que la corrección de los resultados no depende sólo de la corrección de los cálculos realizados para producirlos, sino también del instante en el que éstos están disponibles.

Y para Streaming sin embargo encontramos otro tipo de definiciones, mas enfocadas al ámbito de consumir contenidos digitales desde Internet, por ejemplo:

<sup>3</sup> <https://www.instagram.com/?hl=es>

“La retransmisión (en inglés Streaming), es la distribución digital de contenido multimedia a través de una red de computadoras, de manera que el usuario utiliza el producto a la vez que se descarga. La palabra retransmisión se refiere a una corriente continua que fluye sin interrupción, y habitualmente a la difusión de audio o vídeo.” (wikipedia, s.f.)

Entonces, ¿qué se entiende por el procesamiento en Streaming?

Es el procesamiento entendido como:

- Que se procesa en el instante en que se está produciendo el dato, esto es tiempo real.
- La retransmisión del dato o contenido desde la red social se está produciendo en modo continuo, esto es en Streaming.

Enfocándolo a un ejemplo, sería acceder tecnológicamente a la red social, que está “emitiendo” una señal continua (Streaming), entiendo como tal, los tweets, fotos, entradas del muro, que los usuarios están publicando, y esa “señal” se puede procesar según se está emitiendo (es decir en tiempo real) gracias a las nuevas tecnologías (que son capaces de procesarlas antes de que entre al sistema la siguiente muestra sin que se produzcan problemas de latencia):

En el siguiente apartado se va a detallar cuales son las partes de una arquitectura Streaming.

### **2.2.1. Arquitectura Streaming**

A continuación, se muestra un gráfico con el desglose de lo que consiste una arquitectura Streaming. (Perea, Procesamiento de streams)



Figura 6. Arquitectura Streaming.

### **2.2.1.1. Capa de adquisición y almacenamiento intermedio**

Es el punto de entrada al sistema de procesamiento de streams. Existen diferentes formas de adquisición de los datos:

- Petición/Respuesta. Entre el cliente y servidor se establece comunicación, de modo que el cliente envía una respuesta y el servidor responde.
- Publicador/Suscriptor: Existe la figura de publicador, que “publica” el contenido, y un suscriptor que se “suscribe” a este contenido para recibir el dato. La comunicación es asíncrona, ya que el publicador y el suscriptor no trabajan en comunicación.
- One – way. Un servidor está continuamente enviando el mensaje y no tiene constancia de que el servidor destino lo está recibiendo o no.
- Streaming (socket). En este caso, entre el proceso cliente y el proceso destino se establece un socket para el intercambio de mensajes.

A la hora de realizar o de desarrollar la arquitectura hay que tener en cuenta qué ocurre si la capa de adquisición falla, dado que se está ante un proceso Streaming en tiempo real, lo que puede ocurrir ante un fallo en esta capa es una pérdida de los datos, dado que deja de haber conexión, entre la capa de adquisición y el origen de datos. Para evitar esta pérdida de datos, las

tecnologías permiten realizar un Checkpoint, es decir, grabar periódicamente el log del sistema Streaming, que en caso de existir un error se podría recuperar y tomar como origen de datos.

Otra parte importante a tener en cuenta en esta capa, es que entre la capa de adquisición y la capa de procesamiento puede existir un almacenamiento interno del dato recibido.

### ***2.2.1.2. Capa de procesamiento***

Esta capa, como su propio nombre indica, es la capa donde se produce el procesamiento del dato recibido.

Procesamiento se entiende a que el dato pasa por un algoritmo o una lógica que traduce el dato en otro dato.

Tipos de procesamiento:

- At least One: Al menos una vez. Esta capa se asegura de que todos los mensajes sean procesados al menos una vez, es decir que pudiera darse el caso de que el dato se procesara más de una vez.
- At most one: Cada mensaje se procesa como máximo una vez. Se podría dar la circunstancia de que algún dato no se procese.
- Exactly One: Exactamente una vez. En este caso ningún mensaje se queda sin procesar y ninguno se procesa más de una vez. No todas las tecnologías tienen esta política, por lo que, si se quiere, hay que implementarla.

En función de la capa de adquisición, los datos se procesarán de las siguientes formas:

- Data-in-motion: información que se mueve constantemente como un flujo continuo de datos.

- Data-at-rest: información que está almacenada en disco y es constante durante el procesamiento.
- Continuos query: Los datos pasan a través de una query y se van generando los resultados.

Los algoritmos que se implementan en la capa de procesamiento se enfrentan a diferentes problemas derivados de que el dato se está moviendo y posiblemente cambiando a lo largo del tiempo, por lo que puede pasar que una vez que se ha tratado con datos y se haya llegado a un modelo con ciertas propiedades, dicho modelo ya no valga porque el dato haya cambiado. Además, el flujo de datos puede variar y no ser constante, por lo que se puede dar el caso de que haya que descartar información.

Otro hecho importante en la capa de procesamiento es la ventana de ejecución, es decir la sección de tiempo en la que se realizará el procesamiento. Existen dos ventanas de ejecución:

- Ventana deslizante: se genera el cálculo o el procesamiento en función de la longitud de la ventana de ejecución y del intervalo de cada cuanto tiempo se quiere computar.
- Ventana fija: se realiza procesamiento fijo para una ventana de tamaño fijo a lo largo del tiempo.

#### ***2.2.1.3. Capa de resultados y consulta***

Se trata de la capa en la que se van a almacenar los resultados en un almacenamiento persistente, como puede ser en fichero en disco o en base de datos.

Dado que el flujo de datos es continuo, hay que tener en cuenta que el dato final puede necesitar ser actualizado, por lo que hay que seleccionar bases de datos que permitan dicha actualización.

A continuación, se va a detallar diferentes tecnologías que permiten el adquisición y procesamiento de datos en Streaming.

### **2.2.2. Tecnologías existentes**

#### ***2.2.2.1. Adquisición de datos***

Para realizar la adquisición de datos, se destacan dos tecnologías comúnmente utilizadas: Apache Kafka y Apache Flume.

##### **2.2.2.1.1      *Apache Kafka***

###### **Visión General:**

- Apache Kafka nació en el año 2012, y fue desarrollado originalmente por LinkedIn.
- Kafka es un sistema publicador/subscriptor distribuido, particionado y replicado. Estas características unido a que es muy rápido en lecturas y escrituras lo hacen perfecto para el tratamiento de Streaming. (Ramos, 2014)

###### **Componentes:** (Gracia, Un poco más de Kafka (versión 0.8) , 2013)

- Kafka mantiene los mensajes en categorías llamadas Topics.
- Productores publican mensajes en un Topic Kafka.
- Los consumidores se suscriben a los Topics y reciben los mensajes publicados en estos Topics.

- Kafka se ejecuta como un cluster de uno o más servidores de cada una de los cuales se llama Bróker.

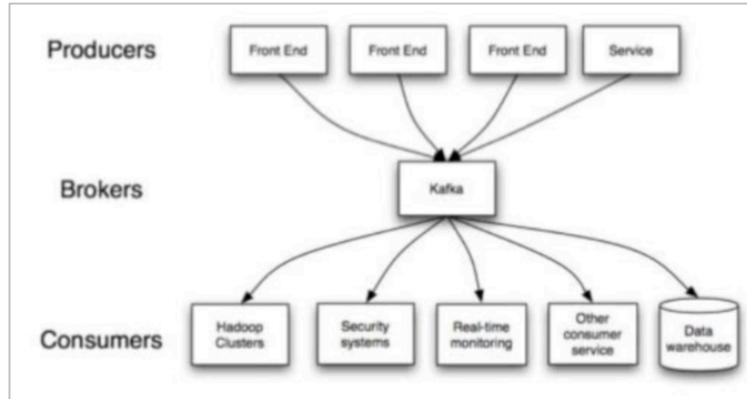


Figura 7. Conceptos Kafka (Perea, Captura y Almacenamiento Intermedio)

### Características principales:

- Rápido: un único bróker de Kafka puede gestionar cientos de miles de megabytes
- Escalable: A través de un cluster escalable los mensajes se pueden particionar en el cluster.
- Durable: Los mensajes se pueden persistir como almacenamiento interno.
- Distribuido: Diseño para ser distribuido lo que garantiza durabilidad y tolerancia a fallos.

Dado que la comunicación entre el suscriptor y el productor es asíncrona e independiente, Kafka permite que el suscriptor esté implementando en otra tecnología, como por ejemplo Spark Streaming.

### 2.2.2.1.2 Apache Flume

**Visión General:** (Flume, s.f.)

- “Servicio distribuido, confiable y disponible para la recolección, agregación y movimiento eficientes de flujos de datos”

**Componentes Principales:**

- Sources (“fuentes”): consumen eventos de cualquier sistema externo y los reenvían a los canales (“channels”)
- Interceptors (“interceptores”): permiten interceptar y modificar eventos al vuelo
- Selectors (“selectores”): permiten definir rutas para los eventos.
- Channels (“canales”): almacenan eventos hasta que éstos son consumidos por un sumidero
- Sinks (“sumideros”): obtienen eventos de un canal para entregarlo/ persistirlo en un destino concreto (normalmente HDFS)

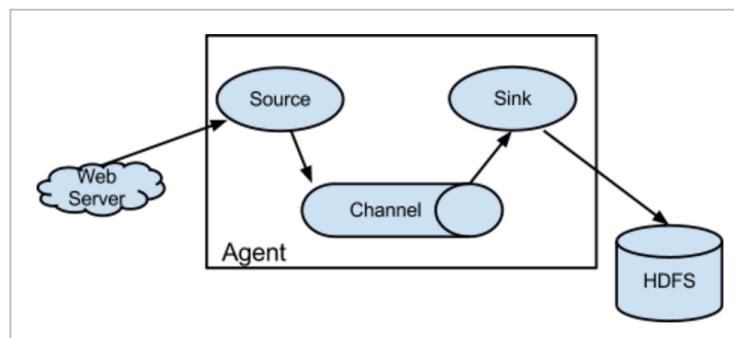


Figura 8. Componentes Flume

**Características principales:**

- Confiabilidad: Los eventos se almacenan en un “channel” hasta que se entreguen a la siguiente fase

- Recuperable: Los eventos se pueden persistir a disco y ser recuperados en caso de fallo
- Declarativo: Los componentes y cómo se enlazan se definen por configuración.
- Extensible: Flume incluye muchas “sources”, “sinks”, etc ya implementados, pero permite añadir otros propios fácilmente.

### ***2.2.2.2. Procesamiento de datos***

Para realizar la adquisición de datos, se destacan las tecnologías comúnmente utilizadas: Apache Storm, Apache Flink y Apache Spark.

#### *2.2.2.2.1 Apache Storm*

##### **Visión General:**

- Apache Storm es un sistema de computación distribuida en tiempo real Open Source.
- Permite procesar streams de datos de manera continua de manera similar a como Hadoop realiza procesamiento por lotes (batch).
- Los streams se tratan como un flujo de datos sobre los que se aplican distintas transformaciones.
- Procesa el dato al menos una vez ( at least one).
- Se puede programar en distintos lenguajes de programación como Scala, Java o Python.

## Componentes:

- Tuplas: Estructura de datos principal de Storm. Es una lista de nombres clave – valor.
- Streams: Una secuencia continua de tuplas que son creadas y procesadas de manera distribuida.
- Spouts: Es la fuente de datos. Leen tuplas de una fuente externa de datos. Pueden emitir más de un stream. Se pueden integrar con Kafka.
- Bolts: Se encargan de la fase de procesamiento. Se suscriben a un Stream de datos.
- Topologías: Grafo de Spouts y Bolts que se conectan mediante una política de agrupamiento. Aquí se incluye la lógica de procesamiento.
- Agrupamiento de streams: Define como el Stream va a ser particionado entre las tareas que se ejecuten en los Bolts.

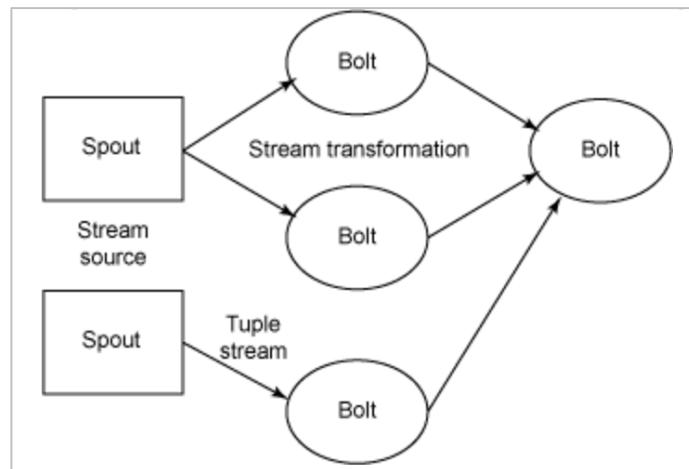


Figura 9. Componentes Apache Storm (IBM, s.f.)

De modo que Spout es el encargado de recoger el flujo de datos de entrada y Bolt es el encargado del procesado o transformación de los datos.

#### **Características principales:**

- Rápido
- Escalable
- Tolerante a fallos
- Garantía de procesamiento de datos
- Fácil de desplegar

#### *2.2.2.2.2 Apache Flink*

#### **Visión General:**

- Es un motor de procesamiento Open Source de flujos de datos en Streaming (y en Batch) distribuido y tolerante a fallos escrito en Java y Scala
- Desarrollo liderado por Data Artisans.
- Procesa el dato exactamente una vez.

#### **Componentes:**

Tal y como se ha dicho, Flink es un framework que se basa en capas:

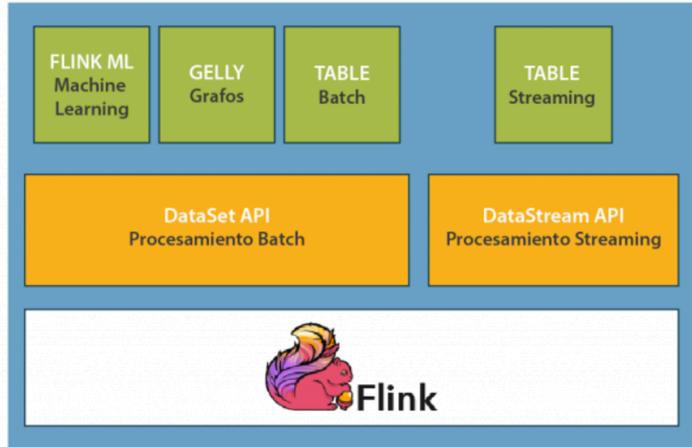


Figura 10. Componentes Flink (Gamboa, Introducción a Apache Flink, s.f.)

De modo de Flink tiene APIS que son las que utilizan el motor de procesamiento de Flink:

- Dataset: representación abstracta que define una colección de datos finita, inmutable del mismo tipo que puede contener datos duplicados.
- DataStream: colección de datos inmutables continua en el tiempo
- Transformaciones: transformaciones de datos de uno o más Dataset/DataStream en uno o varios DataSets/DataStreams.

Para el objetivo que nos ocupa el API que nos interesa sería el DataStream para poder tratar los datos en tiempo real.

#### **Características principales:**

- Tolerancia a fallos: Tiene mecanismos de Checkpoint que garantiza el procesamiento de exactamente una vez.
- Alto throughput y baja latencia
- Modelo de procesamiento continuo de datos basado en control de flujo y operadores persistentes.

- Permite gestionar eventos que lleguen sin orden
- Un único motor de procesamiento para batch y para streaming.
- Permite el cálculo de algoritmos iterativos e incrementales.

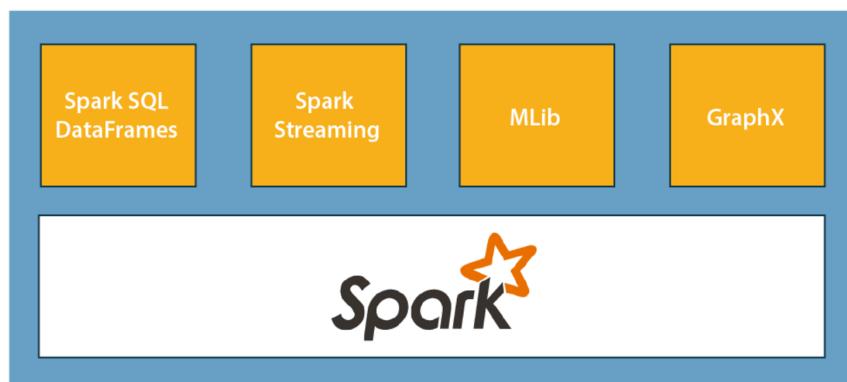
#### 2.2.2.2.3 Apache Spark

##### **Visión General:**

- Creado por Matei Zaharia en UC Berkeley's AMPLab en 2009
- Framework de procesamiento paralelo orientado a Batch y orientado a Streaming, en que se hace uso de operaciones en memoria divididas en varias fases de procesamiento

##### **Componentes:**

Como en el caso de Flink, se trata de un framework de procesamiento con varias capas:



*Figura 11. Componentes Spark (Gamboa, Introducción a Apache Spark – Batch y Streaming, s.f.)*

Para el objetivo que nos ocupa el API que nos interesa sería el Spark

Streaming para poder tratar los datos en tiempo real.

- Spark Streaming, puede ingerir datos de un amplio de fuentes, incluyendo flujos provenientes de Apache Kafka, Apache Flume, Amazon Kinesis y

Twitter, así como de sensores y dispositivos conectados por medio de sockets TCP.

- A grandes rasgos lo que hace Spark Streaming es tomar un flujo de datos continuo y convertirlo en un flujo discreto —llamado DStream— formado por paquetes de datos. De forma que procesa los datos en micro-batches

#### **Características principales:**

- Soporte los diferentes modelos de procesamiento (exactly once, at most once y at least once).
- Utiliza RDDs, que son colecciones de lógicas, inmutables y particionadas de registros que están en memoria.
- Escalable
- Tolerante a fallos (se mantiene Checkpoint de los datos)
- Alto throughput y baja latencia
- Se puede utilizar con gran cantidad de orígenes y destinos.

#### ***2.2.2.3. Capa de almacenamiento***

Para la capa de almacenamiento, existen numerosas alternativas, como guardar en disco (HDFS), o en base de datos. Sin embargo, este proyecto se centra en el almacenamiento en base de datos NOSQL, y para ese planteamiento se presentan dos alternativas: Cassandra y MongoDB

### 2.2.2.3.1      *Cassandra*

#### **Visión General:**

- Originalmente creada por Facebook, ahora esta mantenida por Apache, y es open source.
- Es una base de datos NO SQL, que no tiene esquema
- Modelo de datos de tipo Orientado a Columnas.
- Proporciona drivers con múltiples lenguajes de programación, y un potente conector con Spark.

#### **Componentes:** (Rodríguez)

- Un cluster es un conjunto de nodos o instancias de Cassandra organizados según una tipología.
- Los nodos se comunican entre ellos.
- Cada segundo un nodo se comunica con los otros para saber su estado y localización.
- Un cliente se comunica a través de un driver con el clúster.
- El driver elige a cualquier nodo del cluster que para esa petición que le ha llegado será el nodo coordinador. Lo que quiere decir que no existe punto único de fallo, ya que cualquier nodo del clúster puede coordinar cualquier petición.

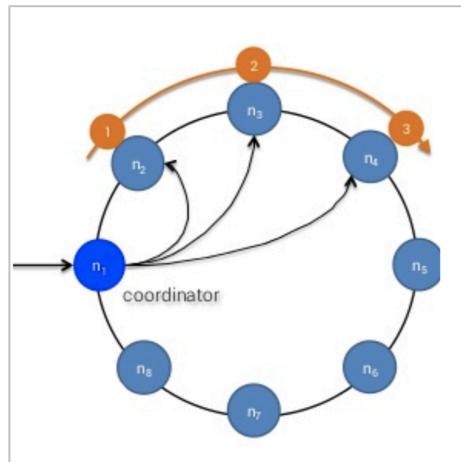


Figura 12. Componentes Cassandra

### Características principales:

- Tolerancia a fallos. No tiene punto unico de fallo.
- Alta disponibilidad
- Elástica, puede crecer tanto horizontal como verticalmente.
- Alto rendimiento en lectura y escritura

#### 2.2.2.3.2 Mongo

### Visión General:

- Es un gestor de base de datos sin esquema
- Modelo de datos orientado a documentos
- Open Source
- Proporciona drivers con múltiples lenguajes de programación

### Componentes:

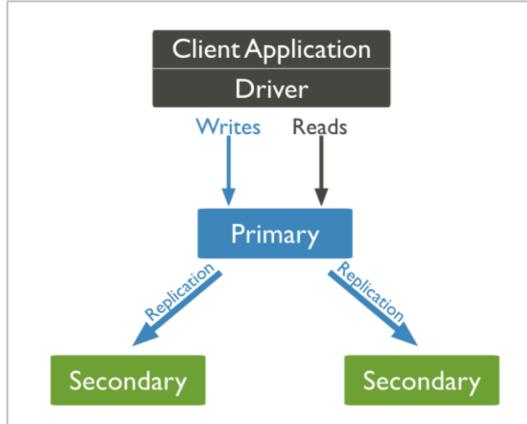


Figura 13. Componentes MongoDB

- Arquitectura basada en Maestro (nodo primario) y esclavo (nodos secundarios).
- ReplicaSet, consiste en un nodo primario y varios secundarios que contienen el mismo conjunto de datos para garantizar la redundancia de datos.
- Las operaciones de lectura y escritura solo se realizan sobre el nodo maestro. Los datos en los nodos esclavos se sincronizan a partir de la copia del nodo maestro.
- Si el nodo maestro cae uno de los nodos esclavos promociona a nodo maestro.

### Características principales:

- Alta disponibilidad
- Prioridad es Consistencia.
- Escalabilidad horizontal
- Replicación nativa
- Utiliza objetos JSON para guardar y transmitir información

Hasta aquí el repaso a alternativas en cuanto a las tecnologías existentes que pueden ser adecuadas para el desarrollo y consecución del objetivo.

### **3. Análisis de alternativas**

Una vez que se ha presentado cuáles son las principales redes sociales existentes y las diferentes tecnologías para realizar el procesamiento de datos en tiempo real, se trata ahora de analizarlas para tomar la decisión posteriormente de cual escoger.

#### **3.1. Red Social**

##### **3.1.1. Diferencias entre Facebook, Instagram y Twitter**

- Las principales diferencias entre Facebook y las demás:
  - Facebook representa la conexión entre amigos y familiares para que un usuario les haga partícipe de sus momentos y gustos personales.
  - Una de las claves de la red es el concepto de privacidad, donde un usuario puede decidir qué publicaciones y qué gustos mostrar en general, a amigos o amigos de los amigos.
  - Facebook proporciona un API para realizar analíticas e informes de actividad sobre las aplicaciones o el muro de un usuario para que analice los accesos que está teniendo. También proporciona una API para extraer la información de los últimos 30 días, sin embargo, no proporcionan un Streaming de datos, y existen restricciones en cuanto al número de llamadas a realizar y en cuanto al tamaño de los ficheros. (Facebook, s.f.)

Desde el punto de vista de las empresas, ellas saben la importancia de tener una página en esta red social, y la tienen para que personas puedan hacerse seguidoras y puedan comentar

acerca de las publicaciones, de forma que personalizan la atención de los clientes consiguiendo una mejor visión de la empresa de cara a los clientes, y por ello subir sus ventas.

- Las principales diferencias entre Instagram y las demás:
  - Esta red social está orientada a que los usuarios comparten fotografías
  - Privacidad, cualquier otro usuario puede seguir a otro y por tanto ver las fotografías que publica.
  - Uso principal es para hacer seguimiento de celebridades, páginas, empresas y poder ver sus publicaciones.
  - En Instagram hay un API de acceso, pero no existen librerías de código para hacer uso de esta API. No es un código que se encuentre extendido entre las diferentes comunidades tecnológicas.

Las empresas están utilizando esta red social para enviar anuncios de sus productos y acercarlos a usuarios.

- Las principales diferencias entre Twitter y las demás:
  - Twitter es la red social de los comentarios en abierto. Como se comentaba antes, es la sala de chat mundial, donde cualquier usuario puede realizar un comentario de cualquier tema y generar un debate alrededor.
  - El tono es diferente a Facebook o Instagram, porque ya no se trata solo compartir o publicar temas personales, como puede ser en el caso de las redes anteriores.
  - Además, dispone de una API y librerías de diferentes lenguajes que proporcionan acceso a la información en Streaming.

- Twitter es la red social utilizada como ejemplo de uso en la mayoría de soluciones tecnológicas, es como el ejemplo de “Hola Mundo” anteriormente utilizado en las tecnologías anteriores.
- Gracias al API de libre acceso, existen numerosos productos comerciales para realizar en estudios sobre Twitter, tales como audiencias, impacto provocado por usuarios influencers, o análisis de sentimientos.

### **3.2. Análisis de las tecnologías**

Como se ha podido ver en el apartado 2.2.2 de Tecnologías para procesar Streaming en tiempo real, existen varias alternativas para la ingesta y procesamiento de datos, de hecho, en Internet se pueden encontrar ejemplos de lectura de datos con Twitter con cada una de ellas.

En general se ha visto que todas las tecnologías tienen en común que son escalables, tolerantes a fallos o altos throughput de lectura con baja latencia. Sin embargo, cada una de ellas tiene sus peculiaridades en función de su arquitectura y lenguajes de programación que haga que la lectura y procesamiento sea más eficaz que otras. En los siguientes apartados se analizará los pros y contras de cada una de ellas en comparación con las otras.

#### **3.2.1. Adquisición de datos**

##### ***3.2.1.1. Apache Flume y Apache Kafka. (Jiang, 2015)***

La principal diferencia entre estos dos sistemas de recolección de datos, es que Flume está pensado para trabajar con Hadoop, con lo que está perfectamente integrado en dicho ecosistema, mientras que Kafka no está diseñado específicamente para Hadoop, sino que Hadoop representa un consumidor más de los que puede tener.

Una de las claves de los beneficios de Kafka es que es muy fácil añadir consumidores sin que ello afecte a la ejecución ni que requiera una parada de servicio. Esto es porque Kafka no revisa si los mensajes han sido o no consumidos. Kafka mantiene un tiempo configurable los mensajes y es el consumidor quien tiene que hacer el seguimiento de los mensajes. En contraste en Flume añadir más consumidores requiere cambiar el fichero de configuración de la tipología para añadirlos, lo que requiere una parada de servicio durante la cual dejarían de recibir los mensajes.

Kafka es capaz de manejar picos altos de llegada de mensajes, amortiguando el impacto entre productores y consumidores. Esto es porque los consumidores son los que recogen los mensajes del tópico, pudiendo recogerlos en diferentes espacios de tiempo. Sin embargo, Flume envía los mensajes a los consumidores, por lo que, si se produce un pico en la llegada de mensajes, los consumidores se pueden ver afectados por el flujo de datos que de repente le están llegando.

En cuanto a la duración de los mensajes, Flume soporta mensajes en memoria y guardar mensajes en disco. Pero en el caso de que el agente de Flume no esté operativo, no podrá acceder a dicho mensajes hasta que se recupere, además no permite guardar datos en nodos diferentes. Sin embargo, Kafka permite la replicación sincronizada o asíncrona según los requisitos del sistema.

El beneficio de Flume es que soporta gran cantidad de conectores de fuentes y de destino, donde un desarrollador sólo tiene que realizar la configuración de la tipología. Sin embargo, en Kafka hay que escribir y desarrollar el código, si bien es cierto que, al ser una

tecnología muy extendida, ya existe código que se pueda reutilizar, pero hay que probarlo y modificarlo para su correcta reutilización.

Otra de las cuestiones en que se diferencian, es que Kafka no hace procesamiento de datos, sin embargo, Flume sí que puede hacer transformación o realizar filtrado con los datos que recibe, lo que hace que al sistema que procesará los datos le lleguen un poco más “limpios”.

### **3.2.2. Procesamiento de datos**

#### ***3.2.2.1. Apache Storm y Apache Flink (Hueske, 2015)***

La principal diferencia entre ambas tecnologías es que Flink es un framework para procesar datos trabajando en Batch y en Streaming, mientras que Storm es un procesador en Streaming sin capacidades Batch.

Ambos son parecidos en la forma de recoger los datos de entrada y tienen en común la baja latencia en el procesamiento, pero Flink ofrece mejores prestaciones, ya que tiene el API de DataStream que contiene las funcionalidades de Map, GroupBy, Windows, Join que en Storm habría que desarrollarlos ad-hoc.

Storm garantiza que el dato esté al menos una vez, mientras que Flink puede garantizar que el dato esté exactamente una vez.

Sin embargo, Flink es una tecnología más nueva que Storm, por lo que, aunque la comunidad de Flink está creciendo más rápidamente, aun se pueden encontrar más código y más experiencia en relación a usuarios en Storm. En la siguiente figura se puede ver el número de repositorios que se pueden encontrar en GitHub de cada una de las tecnologías.

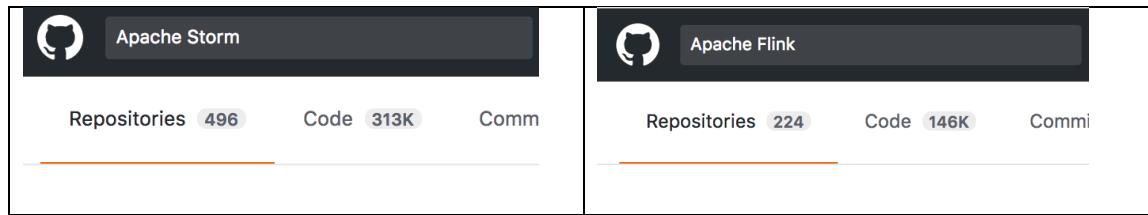


Figura 14. GitHub Apache Storm y Apache Flink

### 3.2.2.2. Apache Storm y Apache Spark (ManojP, 2015)

La diferencia principal entre ambas tecnologías, es que Storm recoge los datos en verdadero tiempo real, mientras que Spark utiliza lo que se llama micro-batching con su API de Spark Streaming.

Aquí se puede ver una comparativa de ambas:

Storm	vs	Spark
• Event-Streaming	Processing Model	• Micro-Batching / Batch (Spark Core)
• At most once / At least once	Delivery	• Exactly Once
• sub-second	Guarantees	• Seconds
• Java, Clojure, Scala, Python, Ruby	Latency	• Java, Scala, Python
• Use other tool for batch	Language	• batching and streaming are very similar
	Options	
	Development	

Figura 15. Comparativa Storm y Spark (Vukelic, 2014)

Spark envía los datos exactamente una vez, sin embargo, en Storm no existe esta opción.

La latencia en Storm es menor que la latencia en Spark.

Trabajar con Spark Streaming dentro del ecosistema de Spark, permite que se trate con los datos en memoria y que se puede realizar procesamiento analítico y machine learning con las otras librerías de Spark mientras se está recibiendo el dato.

Otra de las ventajas que se encuentran en Spark frente a Storm es en lo relacionado con el código, en Spark es mucho más intuitivo, fácil y rápido programar, mientras que, con Storm, hay que programar objetos distintos para ingestar, para procesar y luego definir la tipología. Además, Spark ofrece funcionalidades como Map, agrupamientos, uniones que en Storm hay que programar ad-hoc.

Por último, se muestra el tamaño de las comunidades y repositorios de usuarios de Spark y Storm.

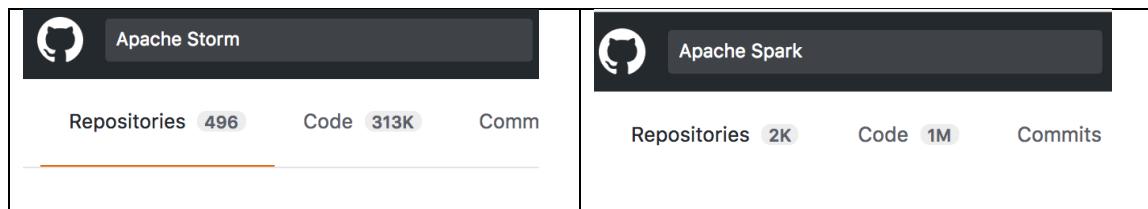


Figura 16. GitHub Apache Storm y Apache Spark

### **3.2.2.3. Apache Flink y Apache Spark (Orlando, 2017)**

Flink al igual que Spark es un framework que permite procesamiento de Streaming y Batch y que proporcionan garantía de procesamiento de exactamente una vez.

La diferencia fundamental de Flink con Spark, es que procesa los datos en tiempo real, es verdadero Streaming, mientras que ya se ha visto que Spark lo hace en micro-batches.

Existen comparativas donde se indica que Flink es más rápido y optimiza mucho mejor los recursos que Spark, por lo que desde el punto de vista de mejorar rendimiento y latencia parece que Flink es una opción mejor (Jacobs, 2016)

A día de hoy, la principal desventaja que tiene Flink es la madurez del producto, de hecho, no tiene una gran publicidad, y además en Machine Learning le faltan muchos algoritmos. Es un producto que está ganando muchos adeptos, pero no llega aún a

encontrarse un gran número de experiencias, y repositorios de usuarios que puedan facilitar el uso y desarrollo de esta tecnología.

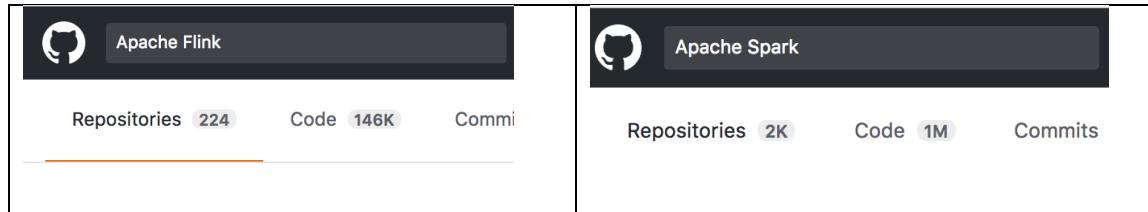


Figura 17. GitHub Apache Flink y Apache Spark

### 3.2.3. Capa de almacenamiento

#### 3.2.3.1. Cassandra y MongoDB (Dharshan, 2016)

Como se ha comentado en el apartado anterior, estas dos bases de datos son NO SQL, sin esquema y ninguna de las dos soporta operaciones ACID.

- MongoDB soporta una modelo de datos orientado a documentos, lo que hace que cada documento tenga propiedades o tipos en niveles. Es un tipo de dato muy orientado a objetos. Cassandra sin embargo presenta un modelo más parecido a las bases de datos tradicional, de tipo filas y columnas, con datos más estructurados que en MongoDB.
- En cuanto a índices secundarios, en Mongo se puede realizar indexaciones por cualquier valor del documento, lo que hace posible que sea fácil realizar consultas por índices secundarios. Sin embargo, en Cassandra no funciona bien los índices secundarios, Cassandra está más centrada en la optimización de consultas basadas en la clave de partición e índices primarios.
- Disponibilidad. El modelo de MongoDB se basa maestro-esclavo, de modo que si un maestro cae uno de los nodos esclavos pasa a ser maestro, durante ese momento

los nodos no están operativos y no se podría acceder a la base de datos para escribir. Cassandra sin embargo no tiene este punto de fallo, no sigue el modelo de maestro-esclavo, por lo que, si un nodo falla, no afecta al resto de escrituras, es por ello que Cassandra asegura la disponibilidad.

- Escalabilidad. MongoDB con su modelo de "maestro único" solo escribe en el nodo maestro, y dejas los nodos esclavos las lecturas. Esto limita mucho la escalabilidad de escritura, ya que solo en 1/3 de los nodos se pueden realizar escrituras. Sin embargo, Cassandra con su modelo de "maestro múltiple" cada nodo puede realizar escrituras, con lo que escala mejor, ya que cuantos más servidores tenga el cluster mejor escalará.
- Cassandra soporta el lenguaje de consulta CQL que es similar a SQL, aunque no permite realizar Select por campos que no estén en la clave primaria, ni hace JOINS, ni soporta funciones. MongoDB en este momento no tiene soporte para un lenguaje de consulta. Las consultas están estructuradas como fragmentos JSON.

## 4. Diseño

Una vez que se han analizado las alternativas de redes sociales, así como diferentes tecnologías para realizar un procesamiento de Streaming, es el momento de tomar la decisión sobre la red social a utilizar, analizar los datos que contiene y realizar el diseño de la arquitectura a implementar.

### 4.1. Selección de Red Social

Tras el breve repaso y análisis a las redes sociales, se determina que Twitter es la red social idónea de la que extraer la información, por las siguientes razones:

- No tiene contenido privado, sino que los usuarios escriben y exponen al mundo sus opiniones y sus preferencias de toda índole. Esto es una característica muy importante si se compara con Facebook, ya que permite poder extraer información asociada a una palabra o hashtag sin filtros por la privacidad.
- Es global. Se accede a Twitter desde cualquier país y en cualquier idioma.
- Por los tipos de comentarios, en Facebook e Instagram, los comentarios están en el ámbito de compartir aficiones o gustos, de hecho es más normal en Facebook comentar con un “Me gusta” en la publicación de otro usuario. Sin embargo la naturaleza de Twitter es distinta, los usuarios entran para comentar, protestar, alabar o criticar acerca de un determinado tema o hashtag del momento.
- Por la facilidad para acceder a los datos, ya que Twitter a diferencia de las otras redes sociales proporciona un API gratuito para el acceso a los datos, sobre el que además se han implementado numerosas librerías en diferentes lenguajes de programación.

#### ***4.1.1.1. Twitter***

Como se ha comentado antes, Twitter es una red social con mensajes limitados a 140 caracteres qué es lo que se conoce como Tweets/Tuits.

##### *4.1.1.1.1 Conceptos de Twitter:*

- Tipos de usuarios:
  - Seguidores(followers): usuarios que se suscriben o siguen los tweets que publican otros usuarios.
  - Seguidos(followed): los usuarios a los que siguen otros usuarios.

De modo que un usuario puede ser seguidor o followers de otro usuario, y así mismo, ser seguido o followed por otros usuarios.

De forma que es una relación asimétrica, dado que unos usuarios no tienen por qué seguir a los otros.

- Hashtag (#):

Un hashtag es una palabra que va precedida por el símbolo #. Se usan para diferenciar, destacar y agrupar una palabra o tópico específico en la red social. De esta forma se pueden agrupar tuits por los que lleven asociado el mismo hashtag en el texto. De hecho, un mismo tweet puede agrupar hashtags diferentes. Los hashtags también se usan para obtener resultados de búsquedas dentro de twitter, ya que al hacer clic sobre un hashtag se pueden obtener tweets similares que contengan el mismo hashtag. (Escudero, 2016)

- Trending Topics

Son los hashtags que más popularidad tienen, es decir, aquellos que la gente más comenta o que incluye en sus tuits.

- Retweet/Retuit

Son los tuits que están referenciados en los tuits de otro usuario.

- Timeline

Es la cronología en el tiempo en que se ven los tuits, entre los que genera un usuario y los que generan los usuarios a los que sigue el usuario. En la siguiente figura se ve un ejemplo de los tuits que vería un usuario al conectarse a la aplicación.

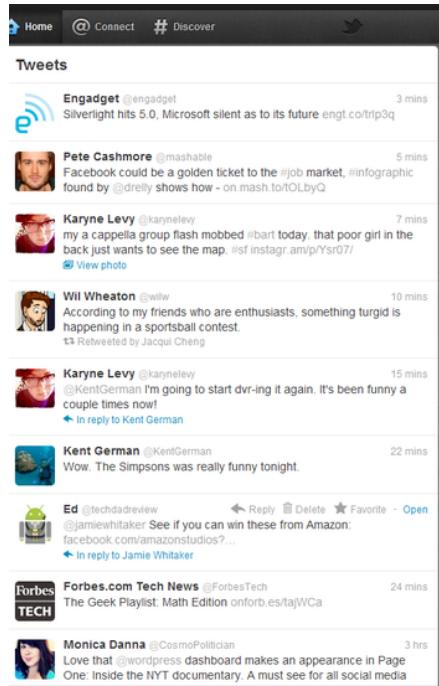


Figura 18. Ejemplo de Timeline (Eje Central, 2016)

## 4.2. Tipos de Tuits

- Convencionales. Son los que están formados simplemente por el mensaje.

Las opciones que se presentan a la hora de crear un tuit son las siguientes:

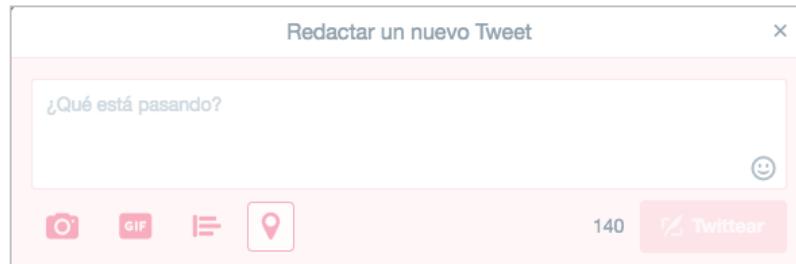


Figura 19 Creación de un tuit.

Se puede insertar texto, foto, un gif animado, emoticonos, una encuesta o la localización.

Ejemplo de un tuit:



*Figura 20. Tuit convencional*

En la figura se puede ver el usuario (@markitoss10), la fecha en que se publicó el tuit, el mensaje y un hashtag que se ha incorporado en el texto (#el hormiguero)

- Tuits Promocionales, que suelen ser usados por empresas:

- Summary Cards (Twitter, Summary Card, s.f.)

Pueden ser usados para muchos tipos de contenido. Sirven para promocionar una web, dando una idea rápida del contenido que pueden visualizar en la web que se incluye en el tuit.



*Figura 21. Tuit Summary card*

En la figura se puede ver que el tuit se incluye un enlace y que twitter muestra automáticamente un fragmento de página web de la url que se inserta en el mensaje.

- Summary Cards Con Imagen (Twitter, Summary Card with Large Image, s.f.)

Son parecidas al caso anterior, la diferencia es que se promociona una web mediante una foto, y es clicando en la foto como se accede a la web que se está promocionando.



Figura 22. Tuit Sumary Card con Imagen

- App Card (Twitter, App Card, s.f.)

Sirven para representar aplicaciones móviles, dando la opción al usuario de instalarla desde Twitter en el terminal móvil. Se presenta el nombre de la aplicación, el ícono, el uso y el precio.



Figura 23. Tuit App Card

- Player Card (Twitter, Player Card, s.f.)

Sirven para mostrar video clips y audio Stream.



Figura 24. Tuit Player Card

#### 4.2.1.1.1 API de Twitter

Como se ha comentado, una de las ventajas de esta red social, es el API de fácil uso y acceso gratuito. En este caso, lo que interesa es API de Streaming, que es la que extrae la información en tiempo real.

Debido a la facilidad comentada, se pueden encontrar librerías en diferentes lenguajes de programación que ya realizan el acceso al API de Twitter.

A continuación, se muestran algunas de librerías existentes según el lenguaje de programación (twitter, s.f.):

- Java:
  - o Twitter4J , creada por [@yusuke](#) — a Twitter API library (Java platform v1.4.2, Android and GAE ready)
- Python:
  - o tweepy maintained , creado por [@applepie & more](#)
  - o twython, creado por [@ryanmcgrath](#)
  - o TwitterAPI, creado por [@boxnumber03](#)
- NET:

- LINQ2Twitter creada por @joemayo (examples)
- Spring.NET Social extension for Twitter by SpringSource —

Esto son solo unos pocos ejemplos de las librerías que se pueden encontrar, pero existe librerías en C, PHP, Curl, JavaScript..

A la hora de escoger una librería, depende mucho de las especificaciones de la aplicación a desarrollar, del lenguaje de programación y de los conocimientos del equipo de desarrollo.

#### ***4.2.1.2. Análisis del dato***

Una vez que ya se ha decidido la red social, y se ha visto el tipo de dato y que existe un API para poder acceder a dicho dato, el siguiente paso es realizar un breve estudio para saber cómo son los datos, la frecuencia, el comportamiento, etc.

##### ***4.2.1.2.1 Volumen de entrada***

Se van a realizar distintas pruebas para analizar el volumen de entrada de Tuits recibidos al que se debe enfrentar el sistema de procesamiento.

- **Hashtag Trendic Topic**

Como se ha explicado anteriormente, trending topic son las palabras clave más utilizadas en un plazo de tiempo concreto en twitter.

Existen numerosas páginas web que muestran cuales son los trending topics del día o del momento.

Ejemplo, el trending topic del dia 19 de Marzo del 2017;

#	Trending Topic	Duración
1	#FelizDiaDelPadre	16:10
2	Chuck Berry	13:50
3	#L6Nirenemontero	11:00
4	#FirstDates287	10:50
5	#gestacionsubrogada	10:30

Figura 25 Trending Topic 19.03.2017 (Trendinalia, s.f.)

Esto quiere decir que, durante 16 horas, el hashtag que más se utilizó en los Tuits de los usuarios fue #FelizDiaDelPadre

Para hacer un estudio de la frecuencia y de los datos que pueden llegar, se ha utilizado la web [http://www\(tweet-tag\).com](http://www(tweet-tag).com) que precisamente sirve para realizar análisis de tuits.

Para el análisis de datos, se ha probado con varios Hashtag en diferentes horarios:

- **#DiaMundialDeLaPoesía** Trending Topic en horario de mañana. El resultado es el siguiente:

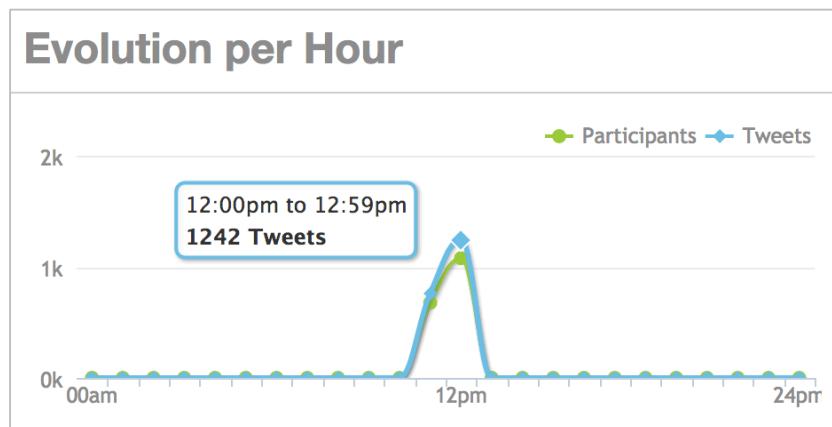


Figura 26 Ejemplo histograma hashtag día ([http://www\(tweet-tag\).com/](http://www(tweet-tag).com/), s.f.)

Entre las 10 y 11 de la mañana se han recibido 761 tuits, y entre las 12 y las 13 horas se han recibido 1.242 tuits.

Utilizando un API de consulta para analizar las entradas relacionadas con dicho hashtag:

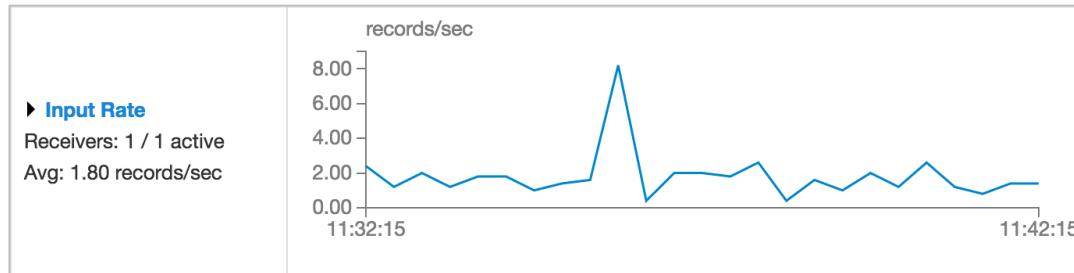


Figura 27. Ratio de entrada hashtag dia

En la figura se puede ver, que este hashtag ha tenido un ratio de entrada de 1,80 tuits por segundo en los diez minutos medidos.

- o #FirstDate289 relacionado con un programa de televisión.

La evolución del numero de tuits que se generan es la siguiente:

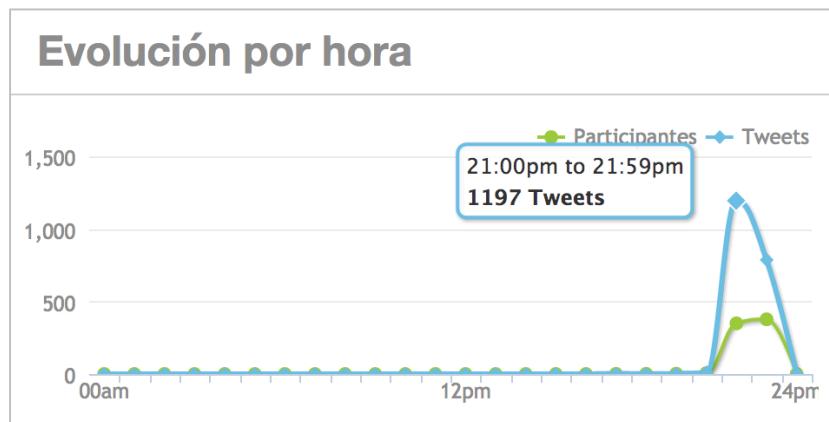


Figura 28 Ejemplo histograma hashtag programa nocturno

Durante la emisión del programa se llegan a ejecutar mas de 2000 tuits, siendo la hora punta entre las 21 horas y las 22 horas.

La estadística del número de tuits por segundo:

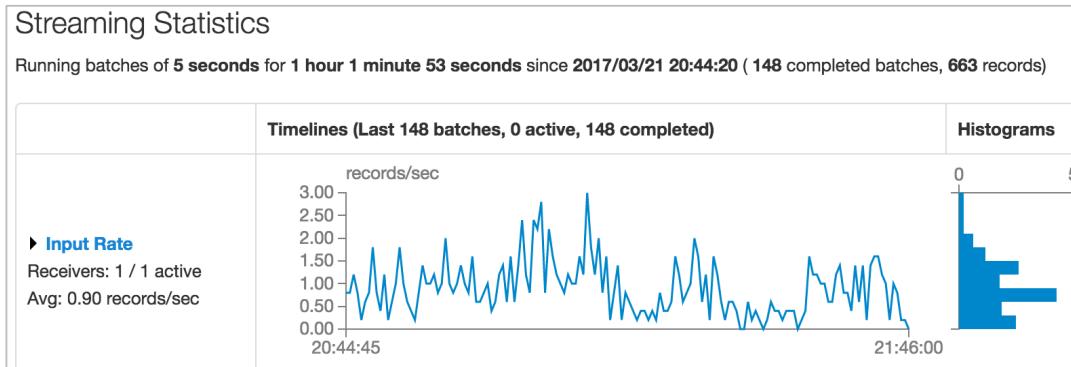


Figura 29. Ratio de entrada hashtag programa nocturno

Una media de casi 1 tuits por segundo, sin embargo se ve en el histograma que hay picos de hasta 3 tuits por segundo.

- Entrada general de tuits

A continuación se muestra el ratio de entrada de todos los tuits que se generan en horario de mañana.

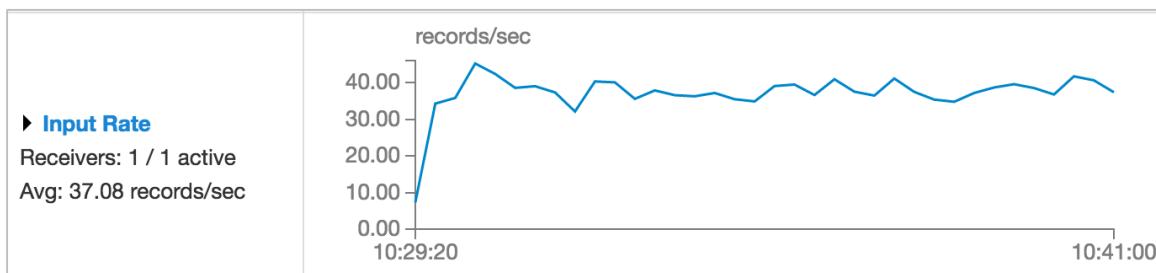


Figura 30.Ratio entrada tuits

En el histograma se puede ver que la media de entrada son 37 tuits por segundo, si bien hay picos que sobrepasan los 40 tuits de entrada.

*Con lo que un dato a tener en cuenta, es que el tiempo de procesamiento de los datos deber ser menor al tiempo en que se ingesta, ya que si no es así se irá produciendo una cola*

*de entrada que ralentizará el sistema y dejará de ser un sistema de procesamiento en “tiempo real”.*

#### 4.2.1.2.2      *Datos en un tuit*

A continuación, se muestra un extracto de los datos más significativo que devuelve el API de Twitter.

*Tabla 1. Objeto Tweet. (Twitter, Tweets, s.f.)*

<i>Dato y Ejemplo del valor</i>	<i>Explicación</i>
createdAt=Wed Mar 22 10:22:50 UTC 2017	Fecha en formato UTC de cuando ha sido creado el tuit.
id=844494572369723392	Identificador único del tuit.
text='Porque tan temprano?'	Texto del tuit.
isFavorited=false	Indica si un usuario ha marcado like sobre el tuit o no.
isRetweeted=false,	Indica si el tuit ha sido retuiteado o no. En un primer momento siempre va a ser falso.
favoriteCount=0,	Indica cuantas veces el tuit ha sido marcado con like por parte de los usuarios.
geoLocation=null, geoLocation=GeoLocation{latitude=40.4203, longitude=-3.7058},	Este campo indica las coordenadas desde las que se posteó el tuit.
place=null,	Cuando este campo tiene valor, se inserta un objeto Place con sus valores.
retweetCount=0,	Número de veces que el tuit ha sido retuiteado
lang='es',	Indica el idioma del tuit, que ha sido identificado con un algoritmo de aprendizaje automático por parte de Twitter. Codificado según el formato BC47.
retweetedStatus=null,	Objeto Tweet. Cuando un tuit es reuiteado de otro, en este campo se inserta todos los valores posibles del tuit que ha sido retuiteado.
User	Objeto Usuario. Los datos del usuario que ha posteado el tuit.

quotedStatusId=-1,	Indica si el tuit ha mencionado otro tuit, en cuyo caso contiene el Id del tuit mencionado.
quotedStatusId=844284985167265792	
quotedStatus=null}	Objeto Tweet. Cuando un tuit ha mencionado otro, en este campo se inserta todos los valores del tuit mencionado.

*Tabla 2. Objeto Place (Twitter, Places, s.f.)*

Dato y Ejemplo del valor	Explicación
name='Madrid'	Nombre del lugar, puede ser del lugar desde donde se postea el tuit, o del usuario.
streetAddress='null',	Dirección. Este campo en caso de estar lleno contiene la localidad, código postal, teléfono, región. etc.
countryCode='ES'	Código del País.
id='206c436ce43a43a3'	Id que representa el lugar.
country='España'	Nombre del País.
boundingBoxCoordinates=[ [Ltwitter4j.GeoLocation;@ 15f42d88],	Las coordenadas del objeto boundingBoxType

*Tabla 3. Objeto usuario. (Twitter, Users, s.f.)*

Dato y Ejemplo del valor	Explicación
id=1237603231	Identificador único del usuario.
name='Guillee',	El nombre de usuario.
screenName='GuilleCorrea',	Alias del usuario
location='General Acha',	La ubicación definida por el usuario para el perfil de esta cuenta
followersCount=356,	El número de seguidores que tiene en ese momento.
friendsCount=343	Indica el número de seguidores de esta cuenta
createdAt=Sun Mar 03 03:39:27 UTC 2013,	Fecha de creación del usuario
favouritesCount=1968,	El número de tuits que este usuario tiene favoritos en la vida útil de la cuenta.

timeZone='Brasilia',	Describe el huso horario de este usuario
lang='es',	El código BCP 47 para el lenguaje de la interfaz de usuario declarado por el usuario
statusesCount=5790,	El número de tuits (incluyendo retuits) emitidos por el usuario.
isGeoEnabled=true,	Cuando es true, indica que el usuario ha habilitado la posibilidad de geolocalizar sus tuits.
isVerified=false,	Cuando es true indica que el usuario ha verificado su cuenta

Como se ha visto Twitter pone a disposición muchos datos con los que poder realizar muchas consultas para llegar a conclusiones en cuanto a un tema.

La clave principal es entender la información que Twitter devuelve en streaming en un tuit. Lo que el API envía es una “foto” de la información actual que tiene un tuit y su usuario, y la información del tuit en cuanto al número de retuits, o favoritos va cambiando a lo largo del tiempo. Así como también cambian los datos relacionados con los usuarios, como por ejemplo los seguidores que tiene.

Es decir:

Momento 0 : se publica un tuit.



Figura 31. Datos de Tuit, momento 0

El API devolverá estos valores:

- favoriteCount=0 y retweetCount=0. Porque en el momento de su publicación estos campos aun no tienen valor.

Momento 1: Alguien marca el tuit como favorito.



Figura 32. Datos de Tuit, momento 1

El API de Twitter **no** va a enviar este dato actualizado, ya que no se trata de una publicación del tuit.

Momento 2 : Alguien retuitea el tuit:

En este momento se modifica la información de esta manera:

El tuit retuiteado

Nuevo Tuit



Figura 33. Datos de Tuit, momento 2

API de Twitter devuelve un nuevo ID relacionado con el nuevo Tuit, pero el campo retweetedStatus contendrá la información del Tuit inicial con los datos de favoriteCount=2y retweetCount=1 de este momento.

Y sería el mismo caso en relación al usuario y los datos de followersCount, friendsCount y favouritesCount, que se van modificando.

*Esto quiere decir, que para hacer un análisis de audiencias en cuanto a la influencia de usuarios o tuits hay que tener en cuenta que la información va siendo modificada con el paso del tiempo, y puede darse el caso que no se tenga el dato de favorito actualizado, ya que ese dato no genera un tuit nuevo y por lo tanto no se envía en el API de twitter.*

#### 4.3. Factores a tener en cuenta.

- Origen, formato de los datos y actualizaciones

Los datos como se ha visto provienen de Twitter, y lo que hace Twitter es enviar el tuit con los datos actualizados en cuenta retuits, favoritos, así como los datos del Usuario actualizado.

Con lo cual, a priori, los datos que cambian y evolucionan con el tiempo son:

- tuit:

- isFavorited
- isRetweeted
- favoriteCount
- retweetCount

La primera vez el tuit que se envía tiene un valor favorito, número de veces que el tuit es retuiteado tiene el valor 0. Sin embargo, cuando un tuit es retuiteado, estos valores cambian y se vuelven a enviar dentro del campo retweetedStatus, Ocurre lo mismo cuando un tuit es mencionado por otro, que en el campo quotedStatus se envía los datos del primer tweet actualizado.

- Usuario:

- followersCount
- friendsCount
- favouritesCount

Cada vez que un usuario envía un tuit, el API de Twitter envía los datos la información del usuario actualizada. Ocurre lo mismo que cuando un tuit es retuiteado, que envía los datos del usuario del tuit actualizados.

Esto quiere decir, que hay que tener en cuenta que la información en el tiempo va cambiando, con lo cual, la arquitectura a diseñar debe tener en cuenta las actualizaciones en el tiempo.

- Latencia / Disponibilidad del dato.

Dado que se trata de procesar y mostrar los datos en tiempo real, y que el sistema origina en Twitter, donde lo interesante es ver qué está pasando con los tuits, lo que la gente está comentando, qué tuits tienen más aceptación, cuáles se retuitean más, etc. Es importante ir mostrando o guardando la información mientras está pasando, donde la latencia máxima necesaria rondaría en torno a los 5 segundos. Menos tiempo sería muy aceptable, pero no sería estrictamente necesario.

- Transformaciones

Dado que se trata de ingerir los datos, y luego persistirlos en una base de datos, será necesario hacer un mínimo de transformación en los datos. De modo que se realizará una selección de campos, ya que hay campos como por ejemplo los datos relacionados con la configuración de colores del Look&Feel del usuario que no tiene sentido guardar. También se van a realizar modificaciones en el tipo de dato, recogiendo datos de tipo numérico y/o fecha y almacenarla como tipo cadena.

- Destino de los datos

Una vez que los datos estén en el destino, se precisa que este destino sirva para lo siguiente:

- Sea el origen de datos de otras herramientas que muestren visualmente el resultado. Por ejemplo, realizar una ingesta de datos desde el almacenamiento a herramientas de visualización como Kibana, Tableau, Graylog o incluso realizar un Look&Feel a medida con JavaScript(D3).
- Sea el origen para realizar un estudio de tendencias acerca de los usuarios, cuales son favoritos, cuales tiene más seguidores, etc.... de modo que se pueda identificar aquellos usuarios que tienen más relevancia, que sean influencers en ciertos temas, que haya que tener en cuenta a la hora de lanzar un producto, servicio o campaña sobre algo.

- Robustez.

El sistema debe ser robusto para poder realizar la ingesta del volumen de datos que Twitter envía, tal y como se ha visto en el punto 4.2.1.2.1. Se ha visto que hay picos de entrada según el momento, el tema y la disponibilidad de los usuarios, ya que hay tramos horarios donde el número de tuits puede llegar a alcanzar los 50 por segundo.

#### **4.4. Diseño técnico.**

Para la realización del diseño, la decisión de la tecnología a emplear se ha iniciado en la capa de procesamiento, y de esta decisión ya han derivado las tecnologías a emplear en las otras capas.

### **Capa de Procesamiento:**

De acuerdo a las necesidades que se han comentado que debe tener el sistema, se decide que el diseño de la arquitectura en cuanto a la capa de procesamiento estará basado en **Spark** y su API de Spark Streaming

Se ha decidido que sea Spark de acuerdo a los siguientes criterios:

- Es un framework robusto, escalable, que representa un ecosistema de APIs que poder utilizar como es Spark Streaming para la ingestión de datos, así como SparkML y SparkSQL, que son necesarios para el procesamiento de datos en cuanto a algoritmos de Machine Learning y acceso con SQL a la información. Como se ha visto Flink también ofrece APIs, pero no están tan maduras y evolucionadas como lo está Spark.
- Aunque la latencia de Spark supone segundos, en vez de microsegundos que ofrecen otros Frameworks, para el caso que nos ocupa, con una latencia de segundos es suficiente para el procesamiento y visualización de la información. Por ello, aunque Flink y Storm tienen latencias menores, sería una precisión que no es necesaria en este caso.
- Garantiza que el dato se envía exactamente una vez, este es uno de los requisitos que se precisan, ya que no se quiere perder ni un solo tuit, y solo se quiere procesar una vez. En este caso, este hecho descarta el uso de Storm.
- Por la potencia de tener los datos en memoria, y al modelo de procesamiento paralelo basado en Ventanas que tiene esta tecnología, que

permite realizar procesamiento de datos con un lote de datos, mientras está recibiendo en otra tarea paralela el siguiente lote de datos.

- Por las funcionalidades ya implementadas que ofrece Spark, como map, reduce, groupby, join, etc. que, en otras tecnologías como Storm, habría que implementar y desarrollar.
- Por la amplia comunidad de usuarios que tiene, de modo que, a la hora de desarrollar el código se hace mucho más rápido y fácil, puesto que en internet se pueden encontrar numerosos ejemplos y resolución de problemas que un desarrollador se puede encontrar.
- Tiene una compañía detrás, Databricks, que está apoyando el uso de Spark, y que ofrece una plataforma de Spark en la que poder desarrollar, con un coste 0 en la instalación y gestión de la infraestructura y el Software.

### **Capa de Adquisición:**

Una vez decidido cuál es la tecnología empleada en la capa de procesamiento, se revisa cuáles son las diferentes opciones que existen para la ingestión de la información desde Spark Streaming.



Figura 34. Sources Data Streaming (Spark, s.f.)

En este caso, dado que Spark Streaming tiene un conector con Twitter, se ha decidido utilizarlo en vez de utilizar otras tecnologías que se han visto como Kafka y Flume.

El motivo de esta decisión:

- Es una librería de Spark, por lo que está optimizada para Spark Streaming, y por ello, irá evolucionando y se le irán incorporando mejoras en la medida en que se hagan para Spark Streaming.
- Asegura la tolerancia a fallos, al igual que lo hace Spark, dado que forma parte del ecosistema.
- Está integrado dentro del mismo sistema Spark, con lo que no hace falta conectar con más sistemas externos intermediarios para la ingesta de datos.
- Es código ya implementado, preparado para funcionar, por lo que no requiere instalación, desarrollo y configuración, que serían necesarios para implementar la ingesta con Kafka y/o Flume.

### **Capa de Almacenamiento:**

En la capa de almacenamiento se requiere una base de datos que tenga las siguientes características:

- Alto throughput en inserciones y lectura de datos.
- Buena conexión con Spark.
- Garantice la disponibilidad.
- Tipo de datos está estructurado de forma columnar.

Por estas razones se ha escogido Cassandra como base de datos para la persistencia de los datos de Twitter, dado que es una base de datos No SQL que garantiza la disponibilidad al no tener punto único de fallo. Además, existe un conector muy potente entre Spark y Cassandra que garantiza su integración.

#### 4.5. Solución técnica.

##### 4.5.1. Diagrama de alto nivel

A continuación, se presenta el diagrama de alto nivel de la solución propuesta.

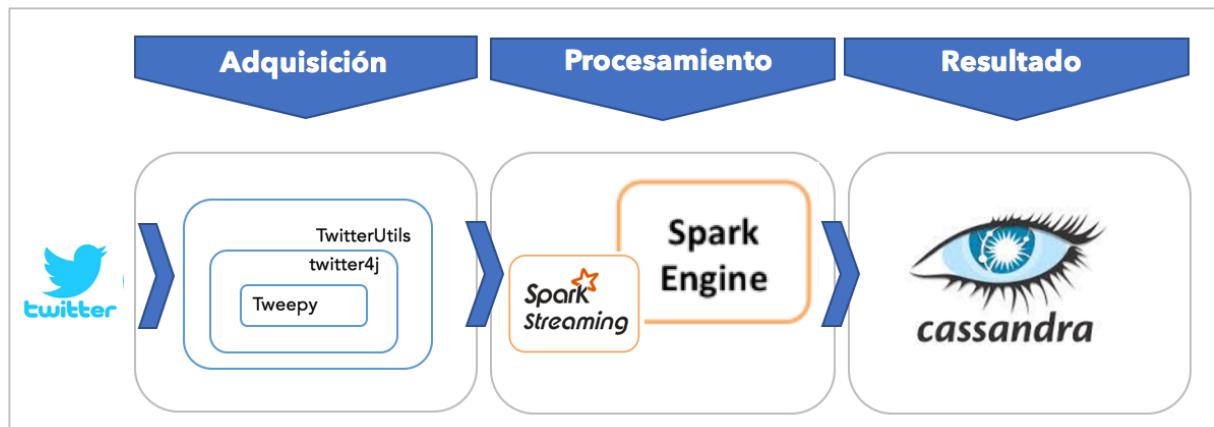


Figura 35. Diagrama alto nivel solución

##### 4.5.2. Código implementado.

A continuación, se detalla únicamente cuál es el código fundamental a desarrollar para la ingesta, procesamiento y resultado de los datos.

El entorno de desarrollo utilizado para la realización de esta prueba de concepto es el siguiente:

- Databricks: Dicha plataforma proporciona gratuitamente un cluster de Spark (versión Spark 2.0 (Scala 2.10)) con 6 GB de memoria.

En dicho entorno, se hará uso de las siguientes APIs:

- spark-streaming-twitter\_2.10-2.0.1: API de acceso a twitter, contiene Twitterutils mencionado anteriormente.
  - spark-cassandra-connector-2.0.0-M2-s\_2.10. API para la conexión con Casandra.
- Portátil MacBook Pro, con las siguientes características:
- versión de SO: OS X El Capitán,
  - Procesador: 2,7 GHz Intel Core i5
  - Memoria: 8 GB
  - Almacenamiento: 250GB

En el cual se ha instalado:

- cassandra-3.9
- kibana-5.2.0
- Logstash-5.1.2
- Elasticsearch-5.2.0
- cassandrajdbc1.1.jar: Api JDBC para que desde Logstash se pueda acceder a Cassandra para la recuperación de datos.

El objetivo del desarrollo para la prueba de concepto de procesamiento de datos en tiempo real es ingestar todos los tuits de idioma español que se producen, actualizarlos y guardarlos en tablas de Cassandra.

La idea es que una vez que esté toda la información en Cassandra, se podría realizar un desarrollo para obtener por ejemplo lo siguiente:

- Trending Topic: Se podrían obtener cuales son los hashtags que más tuits están teniendo.
- Obtener los tuits más retuiteados y analizar cuáles son los hashtags que cuyos tuits tienen más retuits.
- Influencers:
  - o Cuáles son los usuarios con más seguidores
  - o Cuáles son los usuarios que tienen más tuits favoritos.

#### ***4.5.2.1. Adquisición de datos.***

Como se ha comentado en el apartado anterior, para la adquisición de datos se hará uso del API que ya proporciona Spark.

Importación de la librería:

```
import org.apache.spark.streaming.twitter.TwitterUtils
```

*Figura 36. Adquisición Datos. Código 1.*

Configuración de variables de acceso a Twitter:

```
System.setProperty("twitter4j.oauth.consumerKey", apiKey)
System.setProperty("twitter4j.oauth.consumerSecret", apiSecret)
System.setProperty("twitter4j.oauth.accessToken", accessToken)
System.setProperty("twitter4j.oauth.accessTokenSecret", accessTokenSecret)
```

*Figura 37. Adquisición Datos. Código 2*

Y después se utiliza, dentro del contexto de creación del Streaming:

```

| val auth = Some(new OAuthAuthorization(new ConfigurationBuilder().build()))
| val twitterStream = TwitterUtils.createStream(ssc, auth)

```

Figura 38 Adquisición Datos. Código 3

Con estas líneas, se le envían los datos de conexión a Twitter, y luego se genera el Stream de datos desde Twitter.

En este caso, se realiza el acceso sin filtro, para traer todos los tuits. Sin embargo, si se desea se puede filtrar la información de entrada:

```

val twitterStream = TwitterUtils.createStream(ssc, auth, Array("#FelizDia"))

```

Figura 39. Adquisición Datos. Código 4

En este caso, el API solo devolvería los tuits que contengan la cadena #FelizDia en el texto.

#### **4.5.2.2. Procesamiento de datos.**

##### 1) Configuración de Checkpoint

Para asegurar la recuperación ante un fallo, Spark necesita configurar un Checkpoint.

Se trata de un directorio de almacenamiento temporal, donde Spark va almacenando información. Una buena opción es configurar el directorio en un disco distinto a la máquina donde se está ejecutando Spark.

En cualquier caso, lo que se hace es lo siguiente:

- Definir el directorio donde se va a almacenar la información.

```

val checkpointDir = "/checkpoint"

```

- Dentro de la función de proceso de Streaming:

```
ssc.checkpoint(checkpointDir)
```

Se guarda la información generada en el checkpoint.

## 2) Creación y arranque del Streaming.

Para la creación del Stream, es necesario crear o reutilizar el contexto y luego iniciar el Stream. Se utiliza el Checkpoint definido, para que en caso de que el proceso haya de “levantarse” después de un error, utilice los datos almacenados de la ejecución anterior.

```
@transient val ssc = StreamingContext.getOrCreate(checkpointDir, creatingFunc)
```

*Figura 40. Procesamiento, código 1*

Una vez que el proceso se ha lanzado, ya no es posible realizar modificaciones en la funcionalidad implementada en el procesamiento. La funcionalidad y el proceso a realizar está definido dentro de la función **creatingFunc**.

Cuando se detiene el proceso, dejan de recibirse los datos.

En este caso, le estamos indicando que, si existe un activo que lo devuelva y, sino que lo cree.

Se inicia el proceso de Streaming:

```
ssc.start()
```

*Figura 41. Procesamiento, código 2*

El proceso de Streaming se puede parar manualmente, o bien se puede definir cuándo se debe detener.

```
ssc.start()
ssc.awaitTerminationOrTimeout(timeoutJobLength)
```

*Figura 42. Procesamiento, código 3*

Por ejemplo, en este caso le estamos indicando que terminará cuando el proceso termine o bien cuando pase el tiempo en milisegundos definido en la variable timeoutJobLength. En cualquier caso, una vez que se quiere que el proceso finalice hay que pararlo explícitamente:

```
StreamingContext.getActive.foreach { _.stop(stopSparkContext = false) }
```

Figura 43. Procesamiento, código 4

### 3) Proceso de Streaming

En la función CreatingFunc es donde se va a desarrollar las instrucciones de ingesta de los datos.

Una vez creado el Stream, lo que indicamos es el tiempo o intervalo en que se van a ir capturando los datos, es decir, cuanta duración tiene el micro-batching de los datos.

```
val ssc = new StreamingContext(sc, slideInterval)
```

Figura 44. Procesamiento, código 5

En este caso, la variable slideInterval, define los milisegundos en que Spark Streaming está “almacenando” los datos en memoria y generando los RDD’s.



Figura 45. Procesamiento, código 6

En el caso es que slideInterval fuera de 5 segundos, quiere decir que irá generando un DStream de RDD’s, donde cada RDD serán los tuits que han llegado durante esos 5 segundos.

Como se ha visto en el apartado anterior, a continuación, se configura la ingesta de datos:

```
val twitterStream = TwitterUtils.createStream(ssc, auth)
```

Figura 46. Procesamiento, código 7

En este punto, ya se tendría un DStream con tuits que se están generando.

Ahora, se generar una ventana, para procesar los datos en bloque:

```
val twt = twitterStream.window(windowLength, slideIntervalwindowLength)
```

Figura 47. Procesamiento, código 8

Con esta instrucción windowLength le indica el número de segundos que tiene la ventana, y slideIntervalwindowLength indica cada cuantos segundos se debe ejecutar la ventana.

Visualmente:

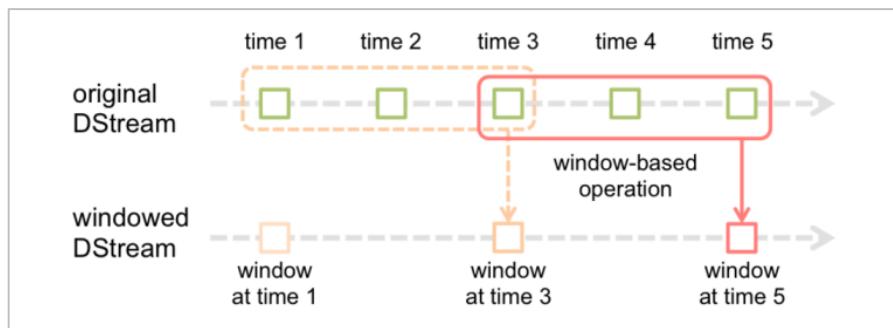


Figura 48. Procesamiento, código 9

Entonces, si se configuran las variables con los siguientes datos:

- slideInterval: 5 segundos
- windowLength: 10 segundos
- slideintervalwindowLength: 10 segundos

Estamos indicando que cada RDD de datos va a contener los tuits que se generan cada 5 segundos, que la ventana va a contener los RDD's que se han ingestado durante 10 segundos

(en este caso serían dos), y que la ventana se va a ejecutar cada 10 segundos (al tener un valor igual al tamaño de la ventana, no se van a configurar solapamientos).

Y una vez se tienen los datos agrupados por ventana de ejecución, ya se pueden procesar con las funcionalidades propias de Spark.

- 1) Filtrado de datos obtener solamente los tuits españoles:

```
val twtCastellan= twt.filter(_.getLang == "es")
```

Figura 49. Procesamiento, código 10

- 2) Mapear y conversión de la información, para dejar solamente los datos de tuits que interesan:

```
val stwt= twtCastellan.map( status=> {
    val id=status.getId()
    val idStr=status.getId().toString()
    val fechaclaveparticion = dateFormat2.format(status.getCreatedAt()).toString() //FORMATO YYYY-MM-DD
    val createdat = dateFormat.format(status.getCreatedAt()).toString()
    val texto=status.getText()
    val usuarioId=status.getUser().getId().toString()
    val nombreUsuario=status.getUser().getName()
    val usuarioSeguidores=status.getUser().getFollowersCount()
    val usuarioAmigos=status.getUser().getFriendsCount()
    val retweetedStatus = status.getRetweetedStatus()
    var tipoTweet="S" // todos son simples al principio.
    //variables para los datos del tweet retweetado
    var rt_idtw = 0L
    var rt_idstr=""
    var rt_fechaclaveparticion =""
    var rt_createdat=""
    var rt_texto=""
    var rt_usuarioId=""
    var rt_nombreusuario=""
    var rt_usuarioseguidores=0
    var rt_usuarioamigos=0
    var rt_favcount =0
    var rt_retweetnum=0
    if (retweetedStatus != null) {
        tipoTweet="RT" // es un retweet.
        rt_idtw=retweetedStatus.getId()
        rt_idstr=retweetedStatus.getId().toString()
        rt_fechaclaveparticion=dateFormat2.format(retweetedStatus.getCreatedAt()).toString()
        rt_createdat=dateFormat.format(retweetedStatus.getCreatedAt()).toString()
        rt_texto=retweetedStatus.getText()
        rt_usuarioId=retweetedStatus.getUser().getId().toString()
        rt_nombreusuario=retweetedStatus.getUser().getName()
        rt_usuarioseguidores=retweetedStatus.getUser().getFollowersCount()
        rt_usuarioamigos=retweetedStatus.getUser().getFriendsCount()
        rt_favcount=retweetedStatus.getFavoriteCount()
        rt_retweetnum=retweetedStatus.getRetweetCount()
    }
    Tweet(id,idStr,fechaclaveparticion,createdat,texto,usuarioId,nombreUsuario,usuarioSeguidores,
          usuarioAmigos,tipoTweet,rt_idtw,rt_idstr,rt_fechaclaveparticion,rt_createdat,
          rt_texto,rt_usuarioId,rt_nombreusuario,rt_usuarioseguidores,rt_usuarioamigos,rt_retweetnum,rt_favcount)
}) //fin del map.
```

Figura 50. Procesamiento, código 11

Una vez que ya se tiene la información formateada, ya se podría procesar como sea necesario, haciendo uso del resto de APIs de Spark, para luego persistir los datos en la base de datos.

#### **4.5.2.3. Capa de almacenamiento.**

Como se ha comentado la configuración y el uso de Cassandra con Spark es muy sencillo, y la integración es muy completa gracias al conector que se ha desarrollado.

##### 1) Configuración de Cassandra en Spark

Para que Spark pueda acceder a la base de datos de Cassandra, hay que configurar un fichero de configuración en el cluster de Spark.

Para ello en el entorno Databricks, lo que hay hacer es:

- Generar el directorio donde se creará el fichero de configuración:

```
dbutils.fs.mkdirs("dbfs:/databricks/init/My Cluster") //My Cluster es el nombre de MyCluster
```

Figura 51. Almacenamiento, código 1

- Crear variables con el nombre del Cluster y la IP de acceso a la base de datos.

```
val sparkClusterName = "My Cluster"
val cassandraHostIP = "89.141.100.1"
```

Figura 52. Almacenamiento, código 2

- Generar el fichero de configuración que debe estar en el directorio de inicio del cluster que Databrick utiliza.

```
dbutils.fs.put(s"/databricks/init/$sparkClusterName/cassandra.sh",
  s"""
    la los las >>
    #!/usr/bin/bash
    echo '[driver].spark.cassandra.connection.host' = "$cassandraHostIP" >> /home/ubuntu/databricks/common/conf/cassandra.conf
    """.trim, true)
```

Figura 53. Almacenamiento, código 3

Una vez que se genera este fichero de configuración, hay que reiniciar el cluster de Spark para que recoja los datos de acceso a Cassandra.

## 2) Creación de tablas en Cassandra.

Dada la naturaleza de los datos de Twitter, se han generado dos tablas para almacenar los datos.

- Tabla tweets: Tabla que almacenará los datos de los tuits.

La consulta que se ha pensado ejecutar sobre esta tabla es con objeto de mostrar todos los tuits que se están produciendo en un día en concreto. Por lo tanto sería una consulta constante del día actual. Esta información ha sido fundamental a la hora de generar la clave de partición y la clave primaria.

Teniendo este dato en cuenta, la tabla se ha diseñado de la siguiente forma:

```
CREATE TABLE kspacetweets.tweets (
    fechclaveparticion text,
    idtw double,
    favcountorig int,
    fechacreacion text,
    hashtag text,
    idstr text,
    nombreusuario text,
    retweetsorig int,
    texto text,
    usuarioid text,
    PRIMARY KEY (fechclaveparticion, idtw)
) WITH CLUSTERING ORDER BY (idtw ASC)
```

*Figura 54. Almacenamiento, código 4*

De modo que la clave de partición es la fecha del tweet en formato YYYY-MM-DD para que todos los tuits del día se almacenen en un nodo, y la clave del registro este formada por dicha fecha y el id identificador del tuit.

- Tabla Usuarios: Tabla que almacenará los usuarios.

En este caso la consulta está pensada para que la consulta sea sobre un usuario en concreto, en cuyo caso la tabla tendría estos campos:

```
CREATE TABLE kspacetweets.usuarios (
    usuarioid text,
    nombreusuario text,
    usuarioseguidores int,
    usuarioAmigos int,
    PRIMARY KEY ((usuarioid )));
```

Figura 55. Almacenamiento, código 6

### 3) Persistencia de los datos en Cassandra.

Una vez que se han procesado los datos, insertar los datos en Cassandra es muy sencillo.

- Importación de librerías.

```
import com.datastax.spark.connector.cql.CassandraConnector
import com.datastax.spark.connector._
import com.datastax.spark.connector.streaming
```

Figura 56. Almacenamiento, código 7

- Cargar los datos en Cassandra.

```
stwt.foreachRDD { rdd =>
    var tablaTweets = rdd.map(TweetTotal=> (TweetTotal.fechaclaveparticion ,TweetTotal.idtw, 0,
                                                TweetTotal.createdat, TweetTotal.idstr, TweetTotal.nombreusuario,
                                                0, TweetTotal.texto,TweetTotal.usuarioid,TweetTotal.hashtag ))
    .saveToCassandra("kspacetweets", "tweets", SomeColumns("fechaclaveparticion", "idtw", "favcountorig",
                                                          "fechacreacion" , "idstr" , "nombreusuario" ,
                                                          "retweetsorig", "texto" , "usuarioid","hashtag"))

    // val tablaUsuarios = rdd.map(TweetTotal=> ("3622415597" , "Karina Fuentes DOS", 500 ,2))

    var tablaUsuarios = rdd.map(TweetTotal=> (TweetTotal.usuarioid ,TweetTotal.nombreusuario,
                                                TweetTotal.usuarioamigos ,TweetTotal.usuarioseguidores))
    .saveToCassandra("kspacetweets", "usuarios", SomeColumns("usuarioid", "nombreusuario",
                                                             "usuarioamigos","usuarioseguidores" ))
}
```

Figura 57. Almacenamiento, código 8

Insertar datos en Cassandra es muy sencillo gracias al conector, basta con salvar el RDD procesado en Cassandra. Con esta instrucción se guardan en Cassandra los nuevos tuits que van entrando. Además, se guardan

los datos del usuario, lo bueno que tiene el conector es que en caso de que el dato del usuario ya existe, lo que hace es actualizar la información en la base de datos que es justamente lo que se necesita.

A continuación, se actualizan los datos de los tuits que han sido retuiteados:

```
//Ahora tratar los retweets que implican la modificación de los datos.
val tweetsRT= stwt.filter(_.tipoTweet == "RT")
tweetsRT.count().map(cnt => "Total : " + cnt + " RT." ).print()

tweetsRT.foreachRDD { rdd =>
    //Ahora pasamos a Cassandra los datos del tweet que el tweet actual ha retuiteado, de forma que si el tweet existe en la
    //tabla ( identificado por la fecha de particion y el id del tweet), machacará la informacion con esta.
    var tablaTweets = rdd.map(TweetTotal=> (TweetTotal.rt_fechclaveparticion ,TweetTotal.rt_idtw, TweetTotal.rt_favcount,
                                                TweetTotal.rt_createdat, TweetTotal.rt_idstr, TweetTotal.rt_nombreusuario,
                                                TweetTotal.rt_retweetnum, TweetTotal.rt_texto,TweetTotal.rt_usuarioid,
                                                TweetTotal.hashtag ))
        .saveToCassandra("kspacetweets", "tweets", SomeColumns("fechclaveparticion", "idtw", "favcountorig",
                                                               "fechacreacion", "idstr" , "nombreusuario",
                                                               "retweetsorig", "texto" , "usuarioid","hashtag"))
    var tablaUsuarios = rdd.map(TweetTotal=> (TweetTotal.rt_usuarioid ,TweetTotal.rt_nombreusuario,
                                                TweetTotal.rt_usuariosamigos ,TweetTotal.rt_usuariosiguidores))
        .saveToCassandra("kspacetweets", "usuarios", SomeColumns("usuarioid", "nombreusuario",
                                                               "usuarioamigos","usuarioseguidores" ))
    //Traza
}

}//fin de bucle foreachRDD
}
```

Figura 58. Almacenamiento, código 9

En estas instrucciones se filtran los tuits que son retuits, para actualizar o insertar en Cassandra los datos del tuit y del usuario retuiteado. De esta forma tan sencilla se actualiza la información en la base de datos de Cassandra.

## 5. Evaluación

Una vez realizada la implementación, se procede a evaluar si la solución es válida.

Se recuerda en este punto cuáles eran los factores que debía cumplir el diseño a implementar:

- Dada la naturaleza de los datos, el sistema debe poder insertar y sobre todo modificar la información, puesto que los datos de twitter pueden evolucionar con el tiempo.

- Latencia del dato, el sistema debe dejar el dato disponible para realizar análisis en 5 segundos. Además, debe poder realizar la ingestión de un alto número de registros por segundo.
- Transformaciones: el sistema debe poder transformar y tratar la información que entra para adaptarla al destino final.
- Destino de los datos: El destino debe contener la información que otras herramientas pueden necesitar para realizar análisis o visualizar la información.
- Robustez. El sistema debe ser robusto para poder realizar la ingestión del volumen de datos que Twitter envía.

Para evaluar estos puntos, se realizan las siguientes evaluaciones:

- inserción y modificación de los datos, para comprobar que los tuits van actualizando los valores según van evolucionando en el tiempo. Así mismo se comprueba que se han realizado transformaciones, dado que el tipo de dato de Twitter se modifica, por ejemplo, en el caso de la fecha de creación o del id del tuit.
- evaluar tiempos de respuesta para comprobar que el sistema puede asumir el volumen de tuits de entrada.
- destino de los datos, se realiza una pequeña demostración de que una vez que se han insertado los datos en Cassandra, se pueden extraer para luego mostrarlos en Kibana.

## 5.1. Inserciones y modificaciones.

Para evaluar esta información, se van a realizar pruebas con un usuario creado para el proyecto, y un hashtag propio que es el que se va a extraer para comprobar la correcta inserción y modificación.

### 1) Inserción de Tuit:

Se genera un tuit:



Figura 59. Evaluación inserción 1

Se extrae y procesa con Spark, y en la base de datos aparece el siguiente registro:

fechaclaveparticion	idstr	favcountorig	retweetsoorig	fechacreacion	hashtag	texto	usuarioid
2017-04-05	84953958690537344	0	0	2017-04-05 10:29:55	#AnaSent	Hoy es día 5 de Abril. Probando Twitter.	819566308924784648

Figura 60. Evaluación inserción 2

en la tabla de Usuarios

usuarioid	nombreusuario	usuarioamigos	usuarioseguidores
819566308924784648	AnaSentTwitt	1	1

Figura 61. Evaluación inserción 3

Como se puede ver en las figuras, se ha insertado los datos del tuit, y los datos de favorito y número de veces retuiteado aparece un 0, lo que es correcto.

### 2) Se modifica la información del tuit, ya que con otro usuario se pulsa en favorito y además retuitea el tuit.



Figura 62. Evaluación inserción 4

Una vez se procesa base de datos ha quedado de la siguiente manera:

idstr	favcountorig	retweetsorig	fechacreacion	hashtag	texto	userid
849539586900537344	1	1	20170405082955	#AnaSent	#AnaSent Hoy es día 5 de Abril. Probando Twitter.	819566308924784640
849541545388900352	0	0	20170405083742	#AnaSent	RT @AnaSentTwitt: #AnaSent Hoy es día 5 de Abril. Probando Twitter.	1710738522

Figura 63. Evaluación inserción 5

La información del primer tuit se ha modificado correctamente, ya que ahora ya aparece que tiene un retuit y un favorito.

Además, se ha almacenado el tuit que ha retuiteado al otro.

En la tabla de usuarios;

userid	nombreusuario	usuarioamigos	usuarioseguidores
819566308924784640	AnaSentTwitt	1	1
1710738522	Araceli Macia Barrad	29	4

Figura 64. Evaluación inserción 6

Se ha insertado el usuario del nuevo tuit.

*Con estas pruebas se verifica que efectivamente se inserta y se actualiza la información correctamente en tiempo real, y que ha funcionado correctamente la transformación realizada en los datos.*

## 5.2. Evaluar tiempos de procesamiento.

A continuación, se va realizar una extracción de datos para comprobar los tiempos de extracción y procesamiento.

La plataforma Databricks donde se está realizando la implementación, cuenta con funcionalidad para poder visualizar los tiempos de ejecución y que es lo que está ocurriendo, lo que es de gran ayuda para evaluar el desarrollo.

1) Ejecución con ventanas de 10 segundos:

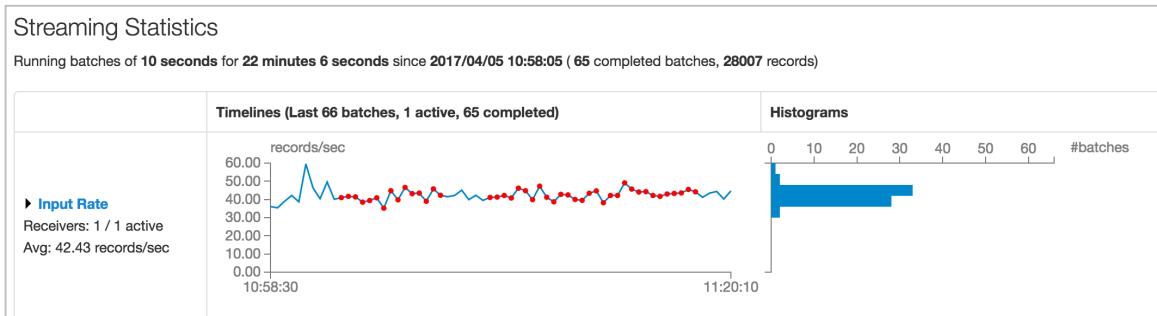


Figura 65. Evaluación tiempos 1

En la ejecución se pude ver que el Batch, es decir el tiempo en que Spark está creando los RDD de tuits es de 10 segundos. Y que, en el tiempo de ejecución, ha procesado una media de 42 tuits por segundo, lo que representan 28007 tuits en total.

El tiempo de procesamiento de cada Batch ha sido el siguiente:

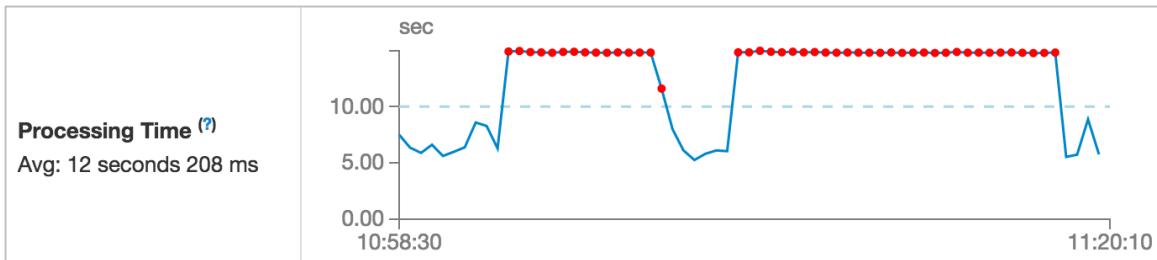


Figura 66. Evaluación tiempos 2

Como se puede ver en la gráfica, el tiempo que tarda en procesar un Batch es de 12 segundos, esto es superior al tiempo en que se están ingestando los datos, lo que produce cuellos de botella, y puede desestabilizar el sistema.

2) Ejecución con ventanas de 5 segundos:

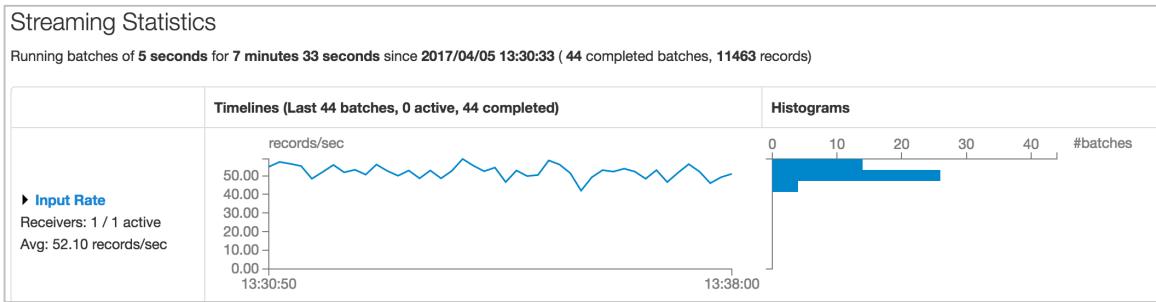


Figura 67. Evaluación tiempos 3

En la ejecución se puede ver que el Batch, es decir el tiempo en que Spark está creando los RDD de tuits es de 5 segundos. Y que, en el tiempo de ejecución, ha procesado una media de 52 tuits por segundo.

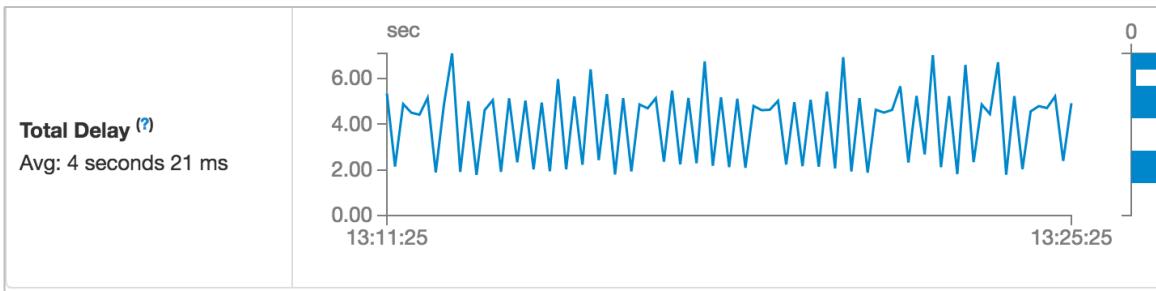


Figura 68. Evaluación tiempos 4

Como se puede ver, ha bajado el tiempo de procesamiento rozando los 5 segundos. Este tiempo no llega a ser perfecto, pero por lo menos no se está colapsado el sistema. Haciendo un análisis del código, puesto que el tiempo de procesamiento es excesivo, se llega a la determinación de eliminar las líneas de código que persisten los datos en Cassandra y se vuelve a probar.

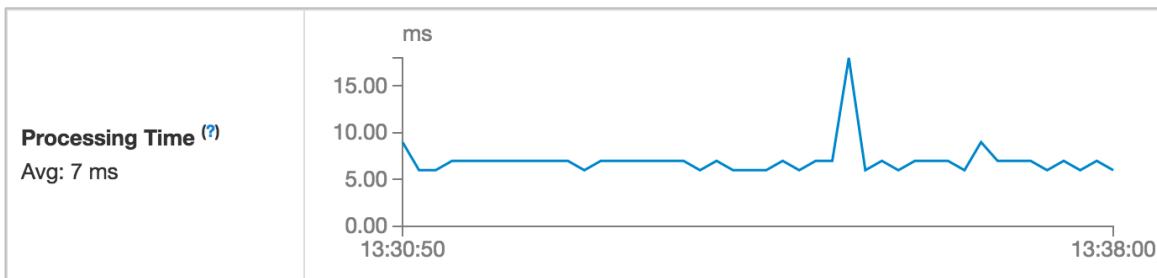


Figura 69. Evaluación tiempos 5

El tiempo de procesamiento ha caído a 7 milisegundos. Lo que quiere decir que el tiempo de procesamiento se está yendo en la conexión con Cassandra.

Esto es normal ya que para realizar las pruebas se ha configurado Cassandra en un portátil, y el tiempo de acceso de Databricks a Cassandra pasa por la velocidad de conexión de la velocidad del operador de telecomunicaciones contratado.

Sin embargo, con este código, y a pesar de estar en un entorno de test no optimizado para un entorno de producción, se da por buena la prueba, puesto que se comprueba que se pueden ingestar un alto número de tuits por segundo y que además el tiempo de latencia es de 5 segundos que es lo que se esperaba.

### 3) Destino de datos:

Como se ha visto en la realización de las pruebas del punto 1, los datos se insertaron en Cassandra correctamente.

Ahora se presenta una pequeña muestra de cómo sería la extracción de datos de Cassandra para mostrar los datos en una herramienta de visualización.

Como herramienta de extracción de datos, se escoge Logstash, que leería los datos de Cassandra mediante la siguiente SQL:

```

select fechclaveparticion, idtw , favcountorig
,fechacreacion ,idstr ,nombreusuario ,retweetsorig
,texto ,usuarioid , hashtag from kspacetweets.tweets
WHERE fechclaveparticion = '2017-04-04' AND
idtw > :sql_last_value ALLOW FILTERING;

```

*Figura 70. Destino de datos, código 1*

En este código se extraen los datos de todas las columnas, utilizando la clave de partición fecha y luego el ID del Tweet. Logstash lo utiliza para almacenar el valor que va leyendo e insertando los datos en ElasticSearch de forma que no se extraigan datos duplicados de Cassandra.

Así, por ejemplo, iría insertando los datos en ElasticSearch:

```

    "idstr" => "849966742411644928",
    "nombreusuario" => "Carmen Quintana",
    "texto" => "RT @raAbf: #ComoMujerMeHaPasado que desde niña escuché como decían a mi madre \"que bien que es niña y podrá cuidarte\"\nY aquí e... ",
    "@timestamp" => 2017-04-06T12:47:30.022Z,
    "fechacreacion" => "20170406124717",
    "retweetsorig" => 0,
    "version" => "1",
    "fechclaveparticion" => "2017-04-06",
    "favcountorig" => 0,
    "usuarioid" => "338568543",
    "idtw" => 849966742411644930,
    "hashtag" => "#ComoMujerMeHaPasado"

```

*Figura 71. Destino de datos, código 2*

Y mediante Kibana, se podría realizar una visualización de los datos que se están procesando en tiempo real:

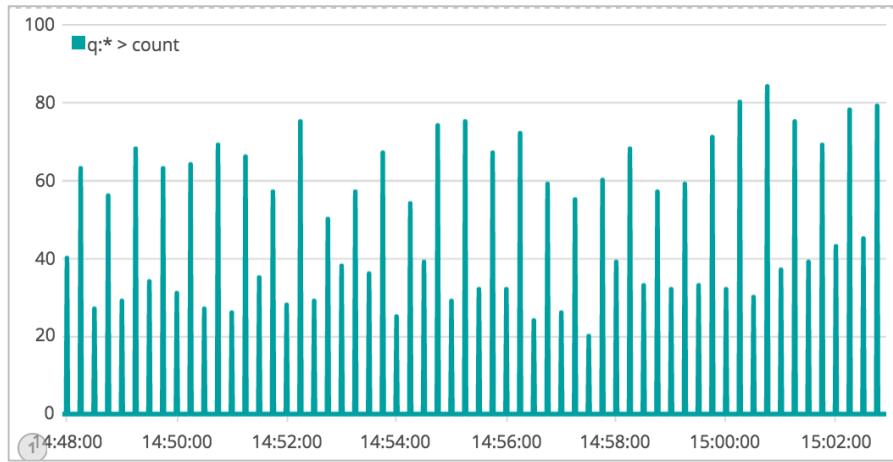


Figura 72. Destino de datos, código 3

En esta figura se pueden ir viendo los tuits que van entrando en general.

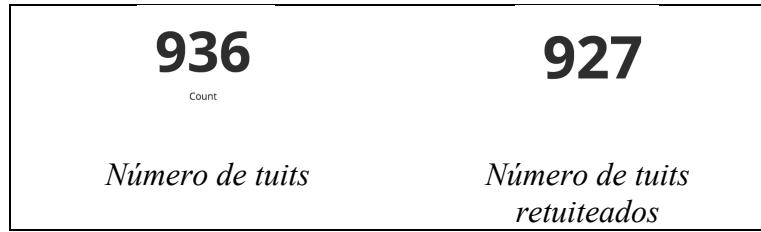


Figura 73. Destino de datos, código 4

Hashtag que se están produciendo:



Figura 74. Destino de datos, código 5

### 5.3. Resumen de la evaluación

A continuación, se muestra el resumen de la evaluación:

Punto a evaluar	Resultado
Inserción y actualización	Correcto. El sistema inserta y actualiza correctamente los datos.
Latencia del dato de 5 segundos	Correcto, el sistema es capaz de procesar y mostrar la información con 5 segundos de latencia.
Transformaciones	Correcto. El sistema puede transformar la información recibida para adecuarla a las necesidades
Destino de datos.	El destino contiene toda la información, y es posible realizar a posteriori análisis y visualizaciones de la información.
Robustez	El sistema es capaz de procesar el volumen de datos que se extraen de Twitter a la velocidad adecuada.

Tabla 4. Resumen de evaluación

El diseño implementado es válido en el entorno de test en que ha realizado el desarrollo de prueba de concepto. Con lo que cuando se escala a un entorno de producción real, el sistema tendrá mejores tiempos de respuesta.

De cara a un producto final real, se tendría que ver con el cliente o empresa, cuáles son las métricas que le interesa analizar y en función de ello, realizar una implementación completa tanto del código de análisis como de los indicadores del cuadro de mando, de modo que la empresa pueda obtener la información que necesita en tiempo real que le dé la ventaja competitiva que se mencionaba en el principio del trabajo.

## 6. Conclusiones

Las empresas se están dando cuenta de que procesar datos en tiempo real y analizarlos promete hacer organizaciones más eficientes y abiertas a nuevas oportunidades. (Anadiotis, 2017). En este proyecto el procesamiento de datos se ha aplicado sobre un caso de negocio basado en una red social, pero hay muchos más ámbitos en el que procesar datos en tiempo real es una gran necesidad. Por ejemplo, en el mundo retail las empresas necesitan identificar las transacciones que se abandonan debido a problemas con tarjetas de crédito en tiempo real. O, por ejemplo, en el IoT, donde se debe procesar la información proveniente de los sensores apenas con microsegundos de latencia.

Por ello, aunque en este proyecto se ha mencionado algunas, en la realidad las tecnologías para hacer procesamiento en Streaming proliferan, con diferentes arquitecturas y sus pros y sus contras.

A modo de ejemplo, en la siguiente figura se puede ver una gran comparativa en cuanto a tecnologías:

<b>Current version</b>	1.6.0	0.6.1	incubating	3.3.0	0.8.0.1+ [available in 0.10]	1.6.1	1.0.0	1.0.0	0.10.0	1.0.2	1.5.0	incubating
<b>Category</b>	DC/SEP	DC/SEP	SEP	DC/ESP	ESP	ESP	ESP/CEP	ESP/CEP	ESP	ESP/CEP	ESP/CEP	SDK
<b>Event size</b>	single	single	single	single	single	micro-batch	single	min-batch	single	single	single	single
<b>Available since [incubator since]</b>	June 2012 [June 2011]	July 2015 [Nov 2014]	[Mar 2016]	Apr 2016 [Aug 2015]	Apr 2016 [July 2011]	Feb 2014 [2013]	Sep 2014 [Sep 2013]	Sep 2014 [Sep 2013]	Jan 2014 [July 2013]	Dec 2014 [Mar 2014]	Sep 2015 [Oct 2014]	[Feb 2016]
<b>Contributors</b>	26	67	19	53	160	838	207	48	159	56	80	
<b>Main backers</b>	Apple Cloudera	Hortonworks	Intel Lightbend	Data Torrent	Confluent	AMPLab Databricks	Backtype Twitter	Backtype Twitter	LinkedIn	dataArtisans	GridGain	Google
<b>Delivery guarantees</b>	at least once	at least once	at least once [with non-fault-tolerant sources]	exactly once	at least once	at least once [with non-fault-tolerant sources]	at least once	exactly once	at least once	exactly once	at least once	exactly once*
<b>State management</b>	transactional updates	local and distributed snapshots	checkpoints	checkpoints	local and distributed snapshots	checkpoints	record acknowledgements	record acknowledgements	local snapshots distributed snapshots [fault-tolerant]	distributed snapshots	checkpoints	transactional updates*
<b>Fault tolerance</b>	yes [with file channel only]	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes *
<b>Out-of-order processing</b>	no	no	yes	no	yes	no	yes	yes	yes [but not within a single partition]	yes	yes	yes *
<b>Event prioritization</b>	no	yes	programmable	programmable	programmable	programmable	programmable	programmable	yes	programmable	programmable	programmable
<b>Windowing</b>	no	no	time-based	time-based	time-based	time-based	time-based count-based	time-based count-based	time-based	time-based count-based	time-based count-based	time-based
<b>Back-pressure</b>	no	yes	yes	yes	N/A	KafkaStream	DSream	Tuple	TridentTuple	yes	yes	yes *
<b>Primary abstraction</b>	Event	FlowFile	Message	Tuple						Message	DataStream	IgniteDataStream
<b>Data flow</b>	agent	flow [process group]	streaming application	streaming application	process topology	application	topology	topology	job	streaming dataflow	job	pipeline
<b>Latency</b>	low	configurable	very low	very low	very low	medium	very low	medium	low	low [configurable]	very low	low *
<b>Resource management</b>	native	native	YARN	YARN	managing e.g. YARN, Mesos, Chronos, Puppet, Salt, Kubernetes, ...)	YARN Mesos	YARN Mesos	YARN Mesos	YARN	YARN	YARN Mesos	integrated*
<b>Auto-scaling</b>	no	no	no	yes	yes	yes	yes	no	no	no	no	yes *
<b>In-flight modifications</b>	no	yes	yes	yes	yes	yes	yes	yes [for resources]	yes [for resources]	no	no	no
<b>API</b>	declarative	compositional	declarative	declarative	declarative	declarative	declarative	compositional	compositional	compositional	declarative	declarative
<b>Primarily written in</b>	Java	Java	Scala	Java	Java	Scala	Clojure	Java	Java	Java	Java	Java
<b>API languages</b>	text files Java	REST (GUI)	Scala Java	Java	Java	Scala Java Python	Java Clojure Python Ruby Node.js Groovy	Java Python Scala	Java	Java Scala Python	Java NET C++	Java *
<b>Notable users</b>	Meebo Sharethrough SimpleGeo	N/A	Intel Levi's Honeywell	Capital One GE Predix PubMatic	N/A	Kelkoo Localsearch AsiaInfo Operable Fairadata Guavus	Groupon Flipboard The Weather Channel Alibaba Baidu Yelp WebMD	Klout GumGum Crowdflower	LinkedIn Netflix Intuit Uber	King Otto Group	GridGain	N/A

Figura 75. Cuadro Comparativo Tecnologías Streaming.

Cuando se tienen los requisitos de lo que tiene que cumplir el sistema a diseñar, y se contrastan con las diferentes tecnologías para llegar a la conclusión de que varias tecnologías cumplen perfectamente con los requisitos que debe cumplir el diseño del nuevo sistema, se llega a la encrucijada de decidir que diseño implementar.

En el caso del sistema expuesto en este proyecto, de extracción de datos de Twitter, otras arquitecturas también habrían sido perfectamente validadas:

- Alternativa 1:

- Adquisición de datos: Logstash con plugin para Twitter.
- Procesamiento: ElasticSearch, que permite cambiar tipos de datos, realizar filtros, crear patrones.

- Almacenamiento: ElasticSearch y visualización con Kibana.
- Alternativa 2:
  - Adquisición de Datos: Apache Kafka, conectado a Twitter.
  - Procesamiento: Storm, para procesar y cambiar tipos de datos.
  - Almacenamiento: Cassandra y visualización con Tableau (con su plugin con Cassandra).
- Alternativa 3:
  - Adquisición y procesamiento de datos: Apache Flink, que también tiene un conector para conectarse a Twitter.
  - Almacenamiento: Cassandra y la visualización en JavaScript con D3.

Con estas alternativas se quiere indicar, que una vez se cumplen los requisitos técnicos como disponibilidad, baja latencia o alto rendimiento, teniendo en cuenta el gran número de alternativas disponibles, hay otras variables que también son importantes y hay que tener en cuenta para la selección de la tecnología.

- Conocimiento y experiencia del equipo de desarrollo. Si el equipo de desarrollo tiene conocimientos en un lenguaje de programación, evaluar si es mejor seleccionar una tecnología con un lenguaje similar que tendrá una curva de aprendizaje menor y por tanto se emplearía menos tiempo en el desarrollo del sistema, o bien, se decide otra tecnología con un lenguaje desconocido con una mayor curva de aprendizaje, que suponga una

inversión en ampliar conocimientos técnicos del equipo e ir rumbo a territorio por explorar.

- Madurez de la tecnología, en cuanto a código, ejemplos, repositorios, experiencias en Internet por parte de otros usuarios, que pueda ayudar mucho a la hora de desarrollar, para minimizar los tiempos aprovechando las experiencias y conocimientos de otros usuarios. Por el contrario, seleccionar tecnologías más innovadoras con menos entradas en internet, donde suponga tarea de investigación por parte del equipo de desarrollo.
- Costes de infraestructura. Determinar la infraestructura que sería necesaria de acuerdo a la tecnología seleccionada, que requisitos de almacenamiento y memoria, y la elaboración de un futuro mapa de crecimiento de acuerdo a las alternativas que ofrezca la tecnología seleccionada.
- Evaluar que supondrían futuros modificaciones o cambios en el sistema.
- Coste de licencia y mantenimiento. Evaluar si se necesita una aplicación con coste, como por ejemplo Tableau, o una aplicación open source, que sin embargo puede tener coste de licencia para tener soporte técnico en caso de problemas.

Una vez que se evalúan todas las variables, con sus ventajas y desventajas, sin perder de vista que toda tecnología es efímera y que por ello hay que estar atento a nuevos avances tecnológicos, se llega al objetivo de diseñar el sistema más óptimo de acuerdo a los recursos que tiene la empresa.

## 7. Referencias

- Alba, J. (2016). *Extractores de Datos*. CIFF.
- Anadiotis, G. (2017). *Streaming hot: Real-time big data architecture matters*. Obtenido de www.zdnet.com: <http://www.zdnet.com/article/streaming-hot-real-time-big-data-architecture-matters/>
- Brandwatch. (8 de Agosto de 2016). *Blog de Brandwatch*. Obtenido de Brandwatch: <https://www.brandwatch.com/es/2016/08/96-estadisticas-redes-sociales-2016/>
- Dharshan. (Agosto de 2016). *Cassandra vs. MongoDB*. Obtenido de <https://scalegrid.io/>: <https://scalegrid.io/blog/cassandra-vs-mongodb/>
- Dotras, E. R. (14 de Noviembre de 2014). *CAMBIO TECNOLÓGICO Y NUEVAS DINÁMICAS DE COMUNICACIÓN*. Obtenido de oikonomics: <http://oikonomics.uoc.edu/divulgacio/oikonomics/es/numero02/dossier/eruiz.html>
- Eje Central. (Febrero de 2016). *El "Time Line" de twitter, cambia*. Obtenido de EjeCentral: <http://www.ejecentral.com.mx/el-time-line-de-twitter-cambia/>
- Escudero, F. (Septiembre de 2016). *Conoce qué son los hashtag en Twitter*. Obtenido de about en español: <http://redessociales.about.com/od/LoBasicoPrimerosPasosEnTwitter/a/Conoce-Que-Son-Los-Hashtags-En-Twitter.htm>
- Est. (s.f.).
- Facebook. (s.f.). *Facebook for Developers*. Obtenido de <https://developers.facebook.com/>: <https://developers.facebook.com/docs/analytics/export>
- Flume. (s.f.). *Apache Flume*. Obtenido de <https://flume.apache.org/>: <https://flume.apache.org/>
- Gamboa, A. H. (s.f.). *Introducción a Apache Flink*. Obtenido de <https://www.adictosaltrabajo.com/>: <https://www.adictosaltrabajo.com/tutoriales/introduccion-a-apache-flink/>
- Gamboa, A. H. (s.f.). *Introducción a Apache Spark – Batch y Streaming*. Obtenido de <https://www.adictosaltrabajo.com/>: <https://www.adictosaltrabajo.com/tutoriales/introduccion-a-apache-spark-batch-y-streaming/>
- Gracia, L. M. (Octubre de 2012). <https://unpocodejava.wordpress.com>. Obtenido de ¿Qué es Apache Flume?: <https://unpocodejava.wordpress.com/2012/10/25/que-es-apache-flume/>
- Gracia, L. M. (Julio de 2013). *Un poco más de Kafka (versión 0.8)* . Obtenido de <https://unpocodejava.wordpress.com>: <https://unpocodejava.wordpress.com/2013/07/10/un-poco-mas-de-kafka-version-0-8/>
- <http://www.tweet-tag.com/>. (s.f.). <http://www.tweet-tag.com/>. Obtenido de <http://www.tweet-tag.com/>: <http://www.tweet-tag.com/>
- Hueske, F. (Junio de 2015). *What is/are the main difference(s) between Flink and Storm? Ask.* Obtenido de <http://stackoverflow.com/>: <http://stackoverflow.com/questions/30699119/what-is-are-the-main-differences-between-flink-and-storm>

- IAB\_EstudioRedesSociales\_2016\_VCorta.pdf. (2016). *Estudio Redes Sociales*. Obtenido de www.iabspain.net: [http://www.iabspain.net/wp-content/uploads/downloads/2016/04/IAB\\_EstudioRedesSociales\\_2016\\_VCorta.pdf](http://www.iabspain.net/wp-content/uploads/downloads/2016/04/IAB_EstudioRedesSociales_2016_VCorta.pdf)
- IBM. (s.f.). *Procese big data en tiempo real con Twitter Storm* . Obtenido de <https://www.ibm.com>: <https://www.ibm.com/developerworks/ssa/library/os-twitterstorm/>
- Jacobs, K. (Agosto de 2016). *Apache Flink: The Next Distributed Data Processing Revolution?* Obtenido de <https://www.data-blogger.com>: <https://www.data-blogger.com/2016/08/13/apache-flink-the-next-distributed-data-processing-revolution/>
- Jiang, L. (Marzo de 2015). *Flume or Kafka for Real-Time Event Processing*. Obtenido de www.linkedin.com: <https://www.linkedin.com/pulse/flume-kafka-real-time-event-processing-lan-jiang>
- ManojP. (Julio de 2015). *Apache Storm vs Spark Streaming*. Obtenido de <https://www.ericsson.com>: <https://www.ericsson.com/research-blog/data-knowledge/apache-storm-vs-spark-streaming/>
- Olmo, L. (2017). *La crisis de United Airlines o cómo internet hunde tu reputación*. Obtenido de www.ticbeat.com: <http://www.ticbeat.com/empresa-b2b/la-crisis-de-united-airlines-o-como-internet-hunde-tu-reputacion/>
- Orlando, L. (Enero de 2017). *Big Data para Javeros con Apache Flink* . Obtenido de <http://equipo.altran.es/>: <http://equipo.altran.es/big-data-para-javeros-con-apache-flink/>
- Pascual, M. G., & Ropero, J. G. (25 de 04 de 2015). *El desafío de una empresa moderna y competitiva*. Obtenido de Cinco Días: [http://cincodias.com/cincodias/2015/04/24/sentidos/1429895580\\_368073.html](http://cincodias.com/cincodias/2015/04/24/sentidos/1429895580_368073.html)
- Perea, I. F. (s.f.). *Captura y Almacenamiento Intermedio*. CIFF, Master en Big Data y Business Analytics.
- Perea, I. F. (s.f.). *Procesamiento de streams*. CIFF, Máster en Big Data y Business Analytics.
- Pilato, D. (Junio de 2015). *Indexing Twitter With Logstash and Elasticsearch*. Obtenido de <http://david.pilato.fr/>: <http://david.pilato.fr/blog/2015/06/01/indexing-twitter-with-logstash-and-elasticsearch/>
- Ramos, J. A. (Octubre de 2014). *Primeros pasos con Apache Kafka*. Obtenido de <https://www.adictosaltrabajo.com>: <https://www.adictosaltrabajo.com/tutoriales/kafka-logs/>
- redindustria. (s.f.). *Definición de tiempo real*. Obtenido de <http://redindustria.blogspot.com.es/>: <http://redindustria.blogspot.com.es/2008/02/definicin-de-tiempo-real.html>
- Rodríguez, J. R. (s.f.). *ESCALABILIDAD EN ALMACENES DE DATOS*. CIFF.
- Rubín, R. (Julio de 2016). *Qué es facebook, cómo funciona y que te puede aportar esta red social*. Obtenido de Ciudadano 2.0: <https://www.ciudadano2cero.com/facebook-que-es-como-funciona/>
- Rubín, R. (Junio de 2016). *Qué es Twitter, cómo funciona y qué te puede aportar esta red social*. Obtenido de Ciudadano 2.0: <https://www.ciudadano2cero.com/twitter-que-es-como-funciona/>
- Spark, A. (s.f.). *Spark Streaming Programming Guide*. Obtenido de <http://spark.apache.org/>: <http://spark.apache.org/docs/2.1.0/streaming-programming-guide.html>

- statista. (s.f.). *Previsión del número de usuarios*. Obtenido de El portal de estadísticas: <https://es.statista.com/estadisticas/474930/redes-sociales-numero-de-usuarios-espana/>
- STRtema1.pdf. (s.f.). <http://www.uco.es/>. Obtenido de <http://www.uco.es/>: <http://www.uco.es/~el1orlom/docs/STRtema1.pdf>
- thecloudavenue. (Marzo de 2013). *Analyse Tweets using Flume, Hadoop and Hive*. Obtenido de <http://www.thecloudavenue.com>: <http://www.thecloudavenue.com/2013/03/analyse-tweets-using-flume-hadoop-and.html>
- Trendinalia. (s.f.). *Trendinalia España*. Obtenido de [www.trendinalia.com](http://www.trendinalia.com): <http://www.trendinalia.com/twitter-trending-topics/spain/spain-170319.html>
- tutorialspoint. (s.f.). *Apache Flume - Fetching Twitter Data*. Obtenido de <https://www.tutorialspoint.com>: [https://www.tutorialspoint.com/apache\\_flume/fetching\\_twitter\\_data.htm](https://www.tutorialspoint.com/apache_flume/fetching_twitter_data.htm)
- tutorialspoint. (s.f.). <https://www.tutorialspoint.com>. Obtenido de Real Time Application(Twitter): [https://www.tutorialspoint.com/apache\\_kafka/apache\\_kafka\\_real\\_time\\_application.htm](https://www.tutorialspoint.com/apache_kafka/apache_kafka_real_time_application.htm)
- Twitter. (s.f.). *App Card*. Obtenido de Twitter: <https://dev.twitter.com/cards/types/app>
- Twitter. (s.f.). *Places*. Obtenido de <https://dev.twitter.com>: <https://dev.twitter.com/overview/api/places>
- Twitter. (s.f.). *Player Card*. Obtenido de Twitter: <https://dev.twitter.com/cards/types/player>
- Twitter. (s.f.). *Summary Card*. Obtenido de <https://dev.twitter.com/cards/types/summary>: <https://dev.twitter.com/cards/types/summary>
- Twitter. (s.f.). *Summary Card with Large Image*. Obtenido de Twitter: <https://dev.twitter.com/cards/types/summary-large-image>
- Twitter. (s.f.). *Tweets*. Obtenido de <https://dev.twitter.com>: <https://dev.twitter.com/overview/api/tweets>
- twitter. (s.f.). *Twitter Libraries*. Obtenido de <https://dev.twitter.com>: <https://dev.twitter.com/resources/twitter-libraries>
- Twitter. (s.f.). *Users*. Obtenido de <https://dev.twitter.com>: <https://dev.twitter.com/overview/api/users>
- Vukelic, D. (Abril de 2014). *Real-Time Streaming with Apache Spark Streaming and Apache Storm*. Obtenido de [www.slideshare.net](http://www.slideshare.net): <https://www.slideshare.net/DavorinVukelic/realtim-streaming-with-apache-spark-streaming-and-apache-storm>
- We Are Social. (2016). *inSlideShare*. Obtenido de slideshare: <https://www.slideshare.net/wearesocialsg/digital-in-2016/427>
- Wikipedia. (26 de 03 de 2003). *Globalización*. Obtenido de Wikipedia: <https://es.wikipedia.org/wiki/Globalizaci%C3%B3n>
- wikipedia. (s.f.). *Streaming*. Obtenido de [es.wikipedia.org](https://es.wikipedia.org): <https://es.wikipedia.org/wiki/Streaming>
- www.expertosnegociosonline.com. (s.f.). *Como usar instagram, que es y para que sirve*. Obtenido de [www.expertosnegociosonline.com](http://www.expertosnegociosonline.com/como-usar-instagram-y-para-que-sirve/): <http://www.expertosnegociosonline.com/como-usar-instagram-y-para-que-sirve/>

