Universitat Oberta de Catalunya

uoc.edu

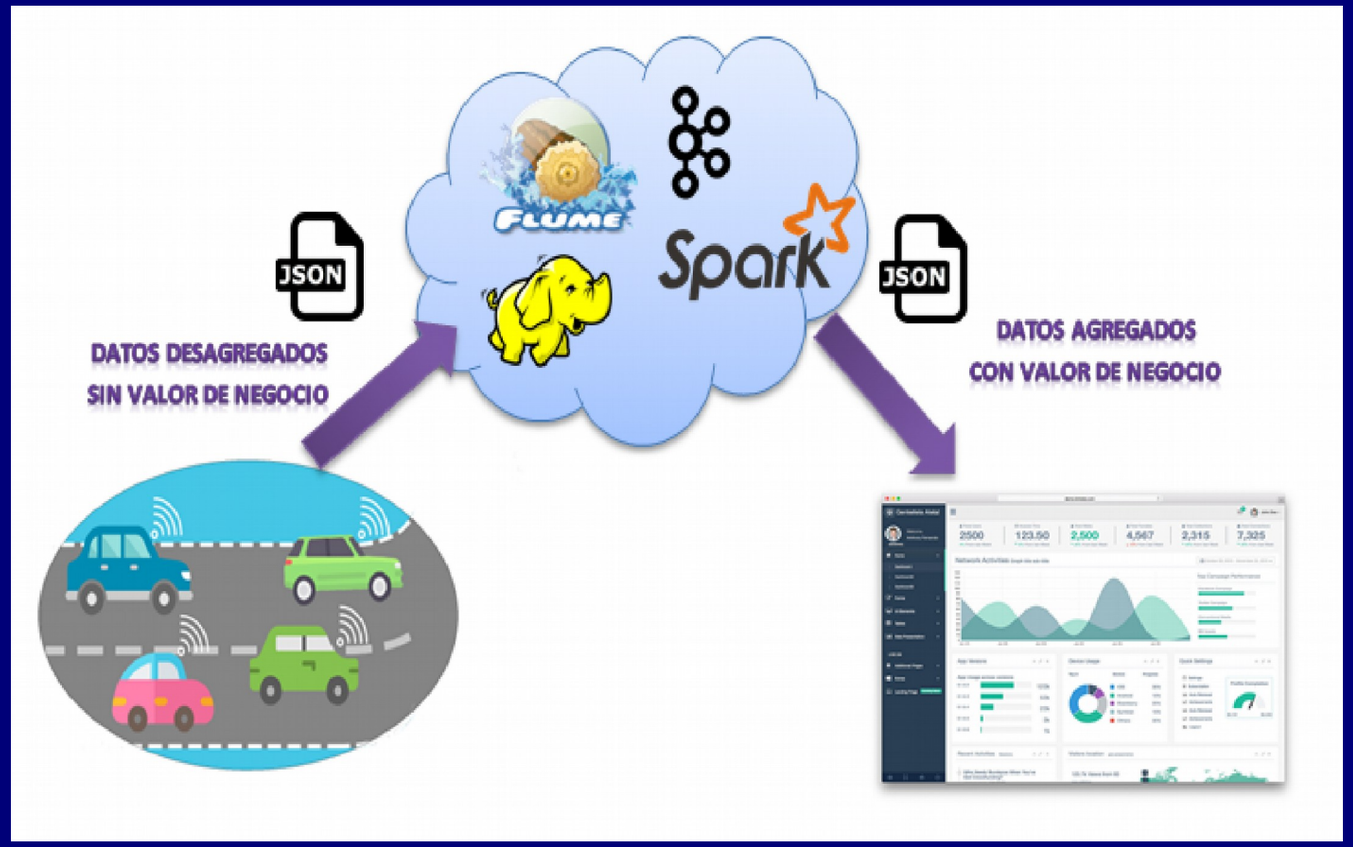
Arquitectura Big Data de Ingesta en Real Time

Autor: Ferran Fernández Garrido
Consultor: David Cabanillas Barbacil
PRA: Josep Curto Díaz

Índice

Problema Planteado

Descripción general



Nuestra problemática, se centra en el procesado de datos en tiempo real, de datos desestructurados JSON sin valor de negocio por si mismos.

Dichos datos son enviados por una empresa ficticia de alquiler de coches y mediante el uso de diversas tecnologías Big Data, debemos poder transformar nuestro input, en una información de valor para ser mostrada en diversas pantallas de visualización de datos.

Casos similares

Netflix y su sistema de recomendación

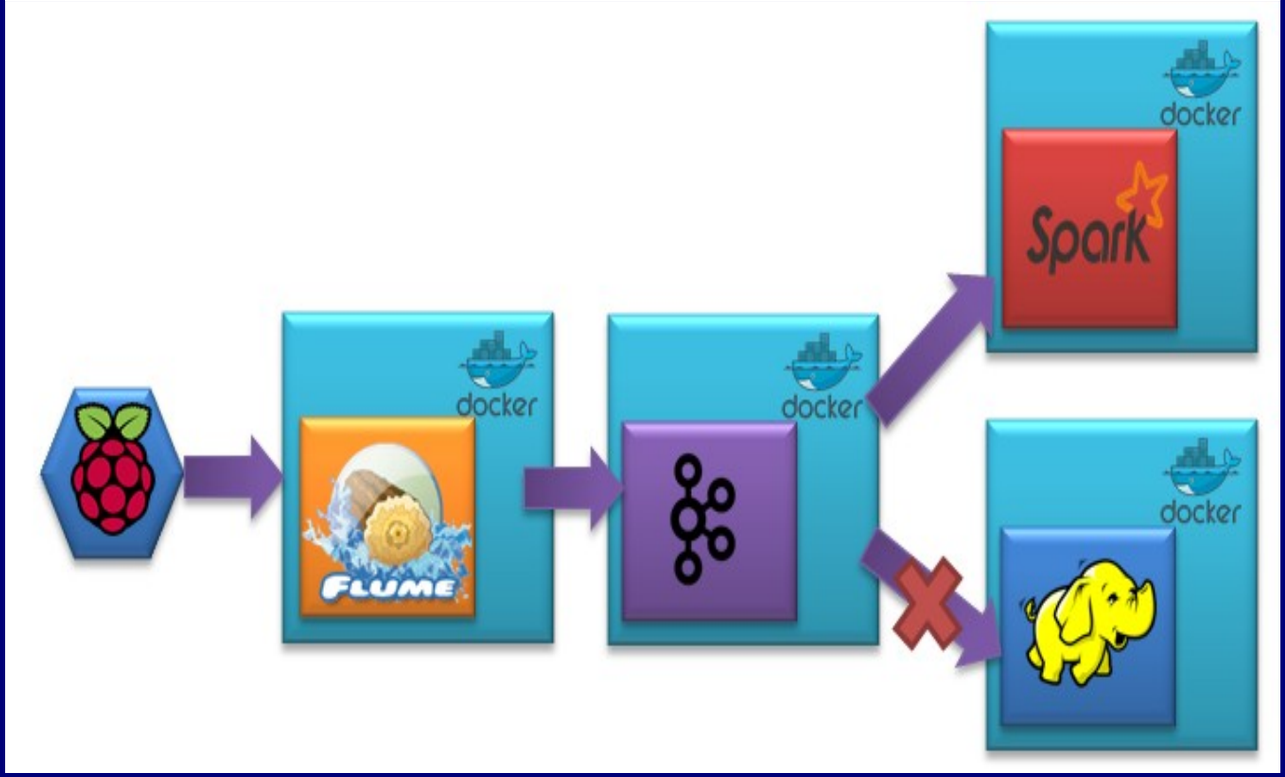
Para encontrar un símil, debemos fijarnos en grandes empresas como Netflix.

Netflix usa un sistema Kafka y Spark Streaming para la recolección de datos de diversas fuentes web. Mediante su uso, son capaces de dar una predicción de gustos y tendencias al usuario mucho más acertada que las de la competencia.

https://www.infoq.com/presentations/netflix-recommendation-spark

Arquitectura propuesta

Diseño a alto nivel



Para este proyecto, se ha usado Docker para contener las diferentes tecnologías Big Data.

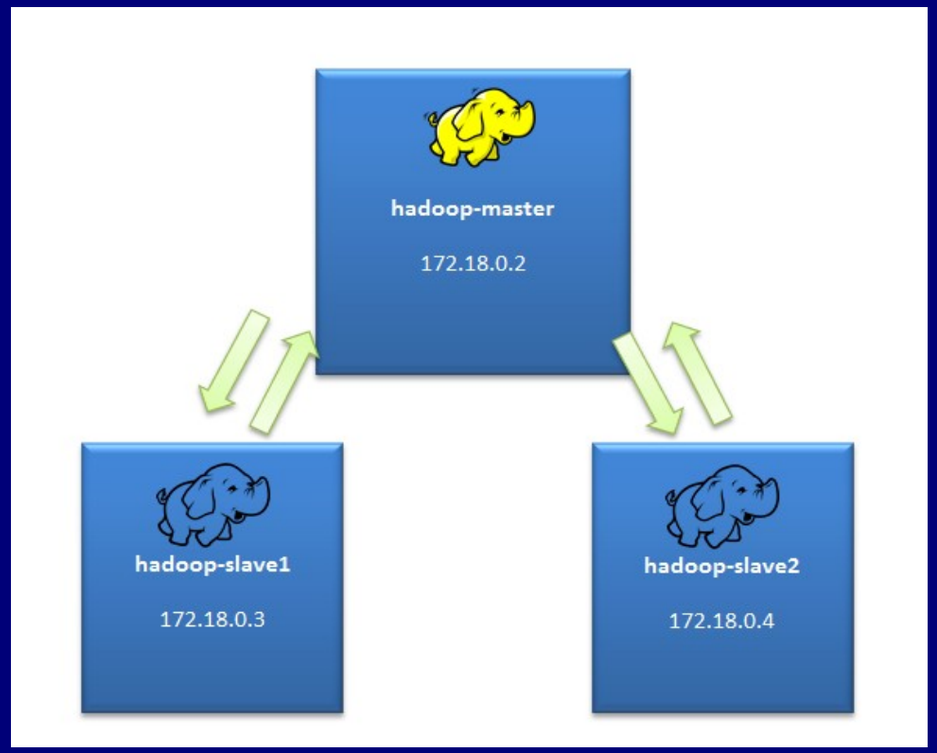
En una descripción de alto nivel, las trazas son enviadas por el sistema Raspberry al sistema Flume.
Desde ahí el siguiente bloque en su recorrido es Kafka donde se encuentran separadas en validas e invalidas.

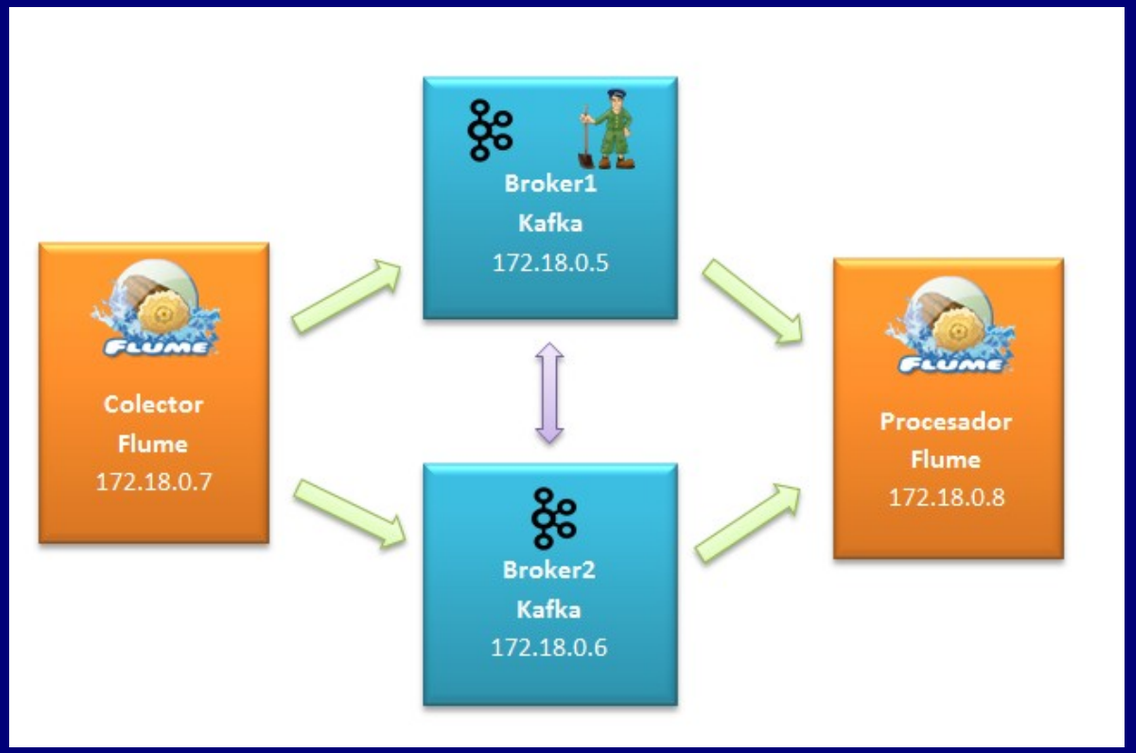
Las invalidas son enviadas a HDFS para su posterior estudio. Las validas por el contrario, se envían a Spark para su agregado y envío de resultados.

Arquitectura propuesta

Diseño a bajo nivel

Las siguientes imágenes muestran la distribución de arquitectura de nuestro sistema Hadoop y Flume + Kafka dentro de los contenedores Docker. Se debe pensar cada recuadro como una máquina diferenciada.

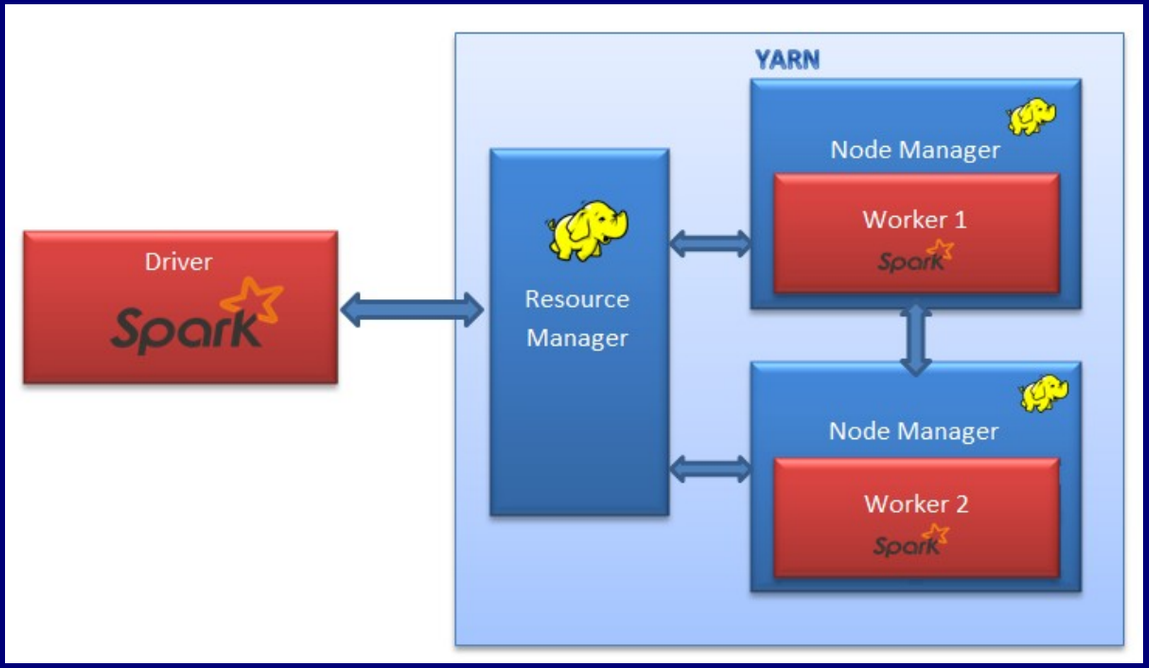


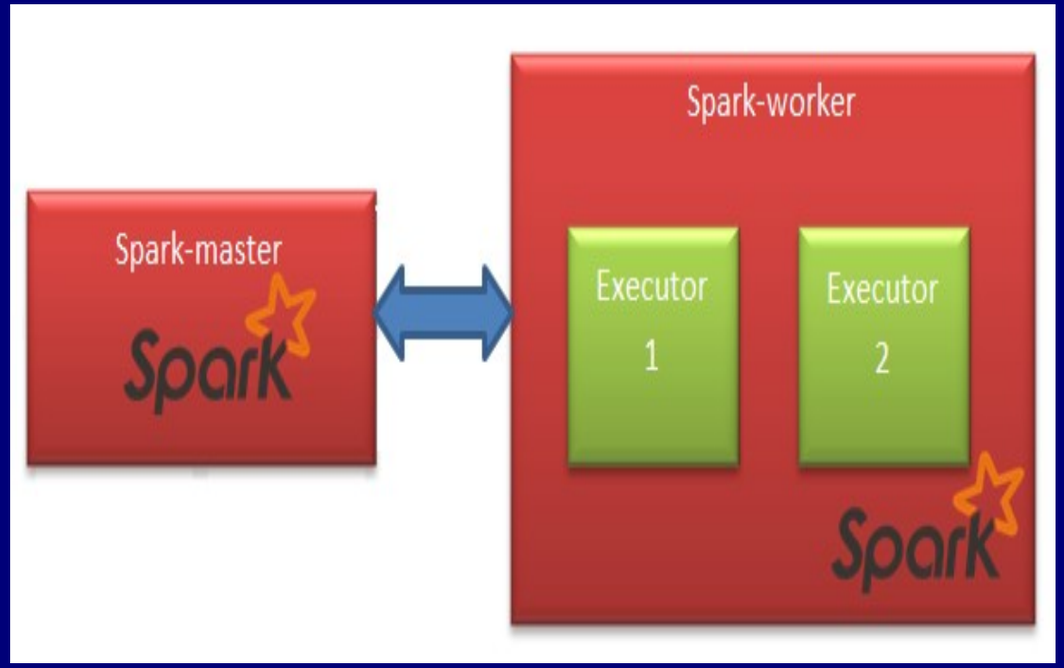


Arquitectura propuesta

Diseño a bajo nivel

Las siguientes imágenes muestran la distribución de arquitectura de nuestro sistema Spark. La imagen de la izquierda muestra el lanzamiento en modo cluster YARN y el de la derecha en modo standalone.





Arquitectura propuesta

Desarrollo en el proyecto

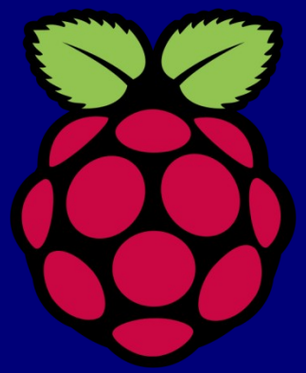
Además de todas las configuraciones y creaciones de scripts, los desarrollos propios de este proyecto son los siguientes:

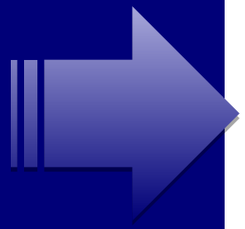
Interceptor: Módulo propio de Flume que se ha creado mediante las librerías que aporta Flume y que sirve como primer punto de descarte de mensajes. Causa un efecto de Multiplexación.

Proceso Spark: Se ha generado un modelo que se mapea mediante Jackson y que se procesa mediante la API de Spark Streaming + Kafka. Los agregados son propios del proyecto sin embargo la lógica está paquetizada en su gran mayoría por lo que seria fácil ser adaptable para otros proyectos Spark Streaming.

Diseño y pruebas de rendimiento

Diseño de envio






```
[
  {
    "headers": {
      "timestamp":"30-05-2017 10:54:21",
      "host":"gtu0091013"
    },
    "body":"
    {\"id\":\"gtu0091013\",\"model\":\"Ford\",\"sta
    tus\":\"running\",\"speed\":3,\"coordinates\":
    {\"long\":\"41.476550\",\"lat\":\"2.251875\"},\
    "tech_features\":
    {\"engine_status\":\"ok\",\"gasoline_level\":49
    ,\"battery_status\":\"ok\",\"oil_status\":\"ok\
    ",\"breaks_status\":\"ok\",\"preasure_level\":
    {\"upper_left\":\"ok\",\"upper_right\":\"ko\",\
    "down_left\":\"ok\",\"down_right\":\"ok\"}}}"
  }
]
```


Las trazas son enviadas desde la Raspberry a Flume mediante el uso del script **mainSend.php.**

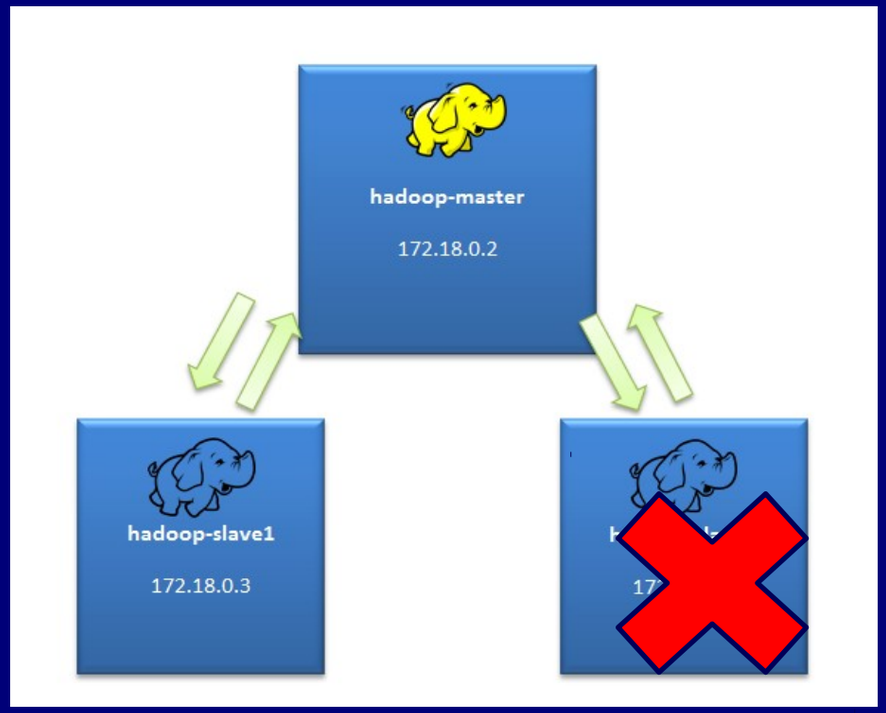
El script genera trazas aleatorias en el formato esperado por flume introduciendo errores con un factor aleatorio (para hacerlo más realista).

Diseño y pruebas de rendimiento

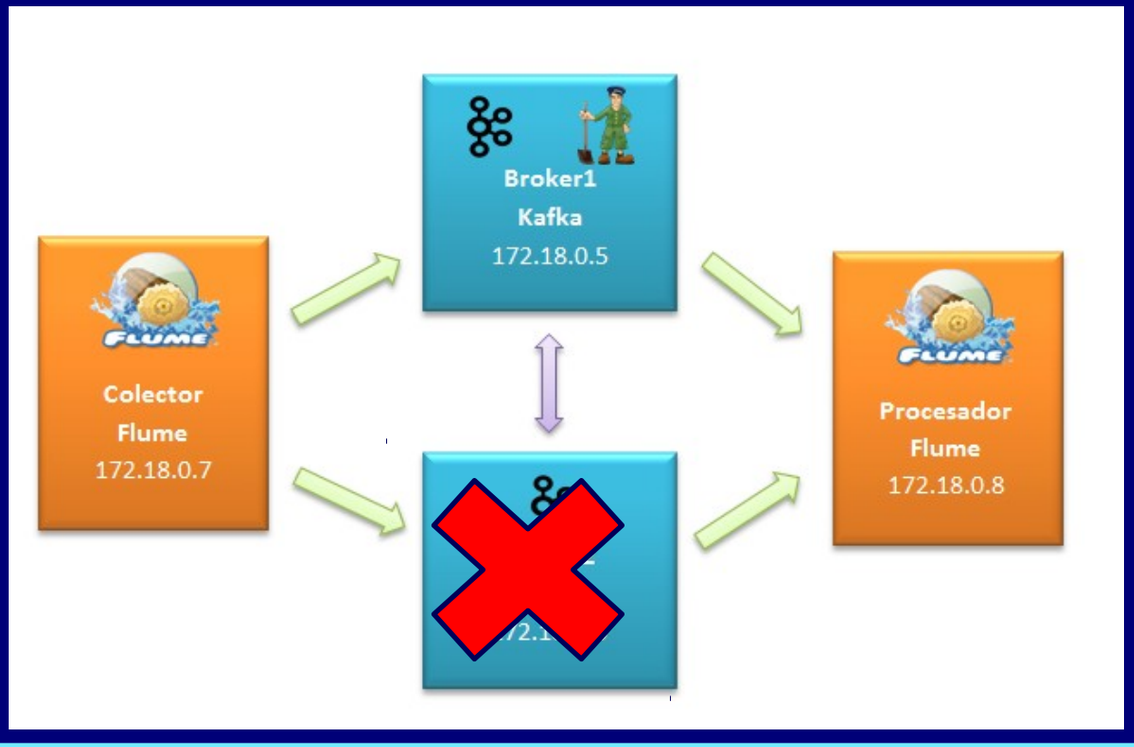
Casos de prueba 1 y 2

Para probar nuestra arquitectura, se crearon casos de prueba para testear el funcionamiento del sistema general:

Caso 1: Nodo HDFS caído



Caso 2: Broker Kafka caído

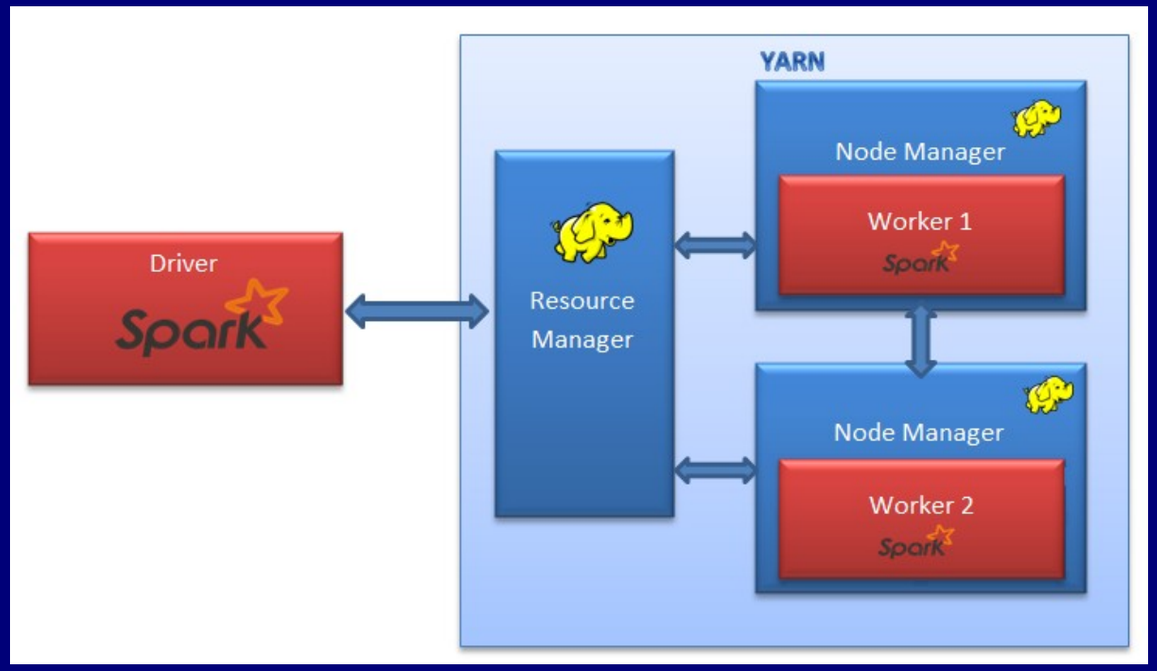


Diseño y pruebas de rendimiento

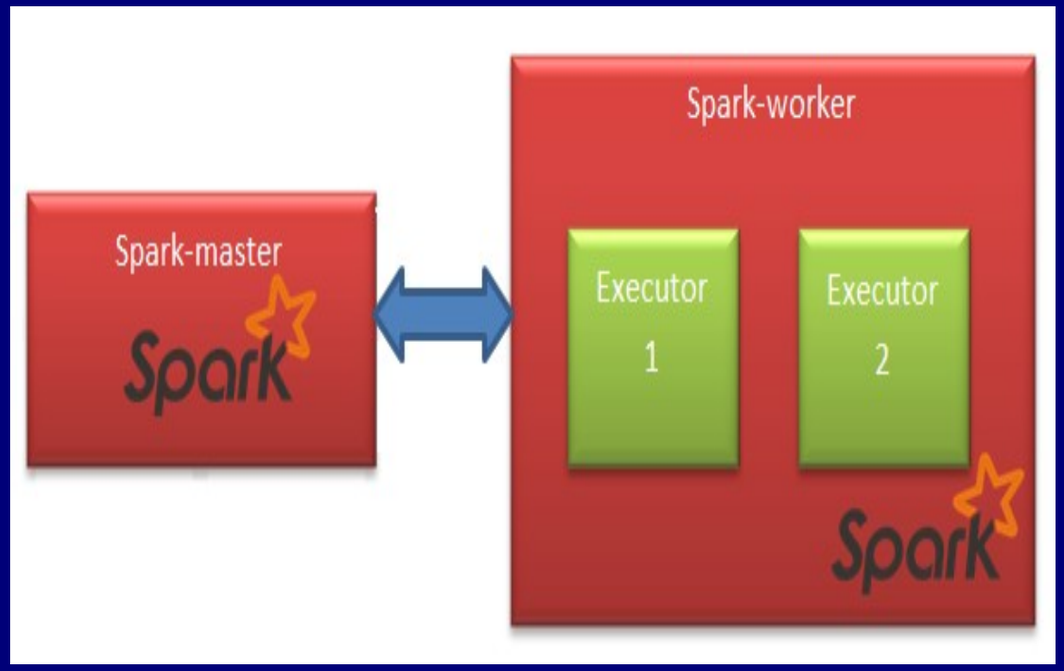
Casos de prueba 3 y 4

Para probar nuestra arquitectura, se crearon casos de prueba para testear el funcionamiento del sistema general:

Caso 3: Spark en modo YARN



Caso 4: Spark en modo standalone



Resultados

Resultados casos de prueba

- Caso 1. Nodo HDFS caído: No se apreció deterioro alguno del sistema puesto que el nivel de escritura de trazas incorrectas es bastante inferior al número de trazas correctas.

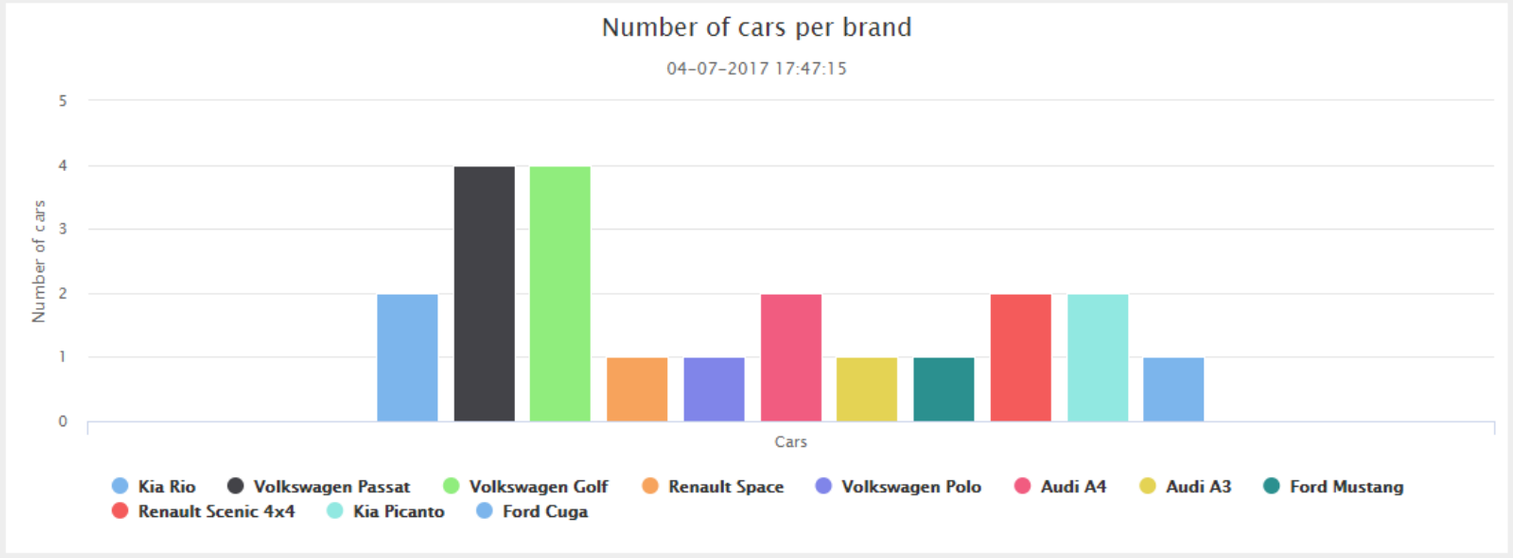
- Caso 2. Broker Kafka caído: No se apreció deterioro general del sistema puesto que la fuente de datos no era capaz de enviar a una cadencia de datos tan grande como para notar un deterioro de condiciones.

- Caso 3. Spark en modo YARN: Debido a que requiere de un gran volumen de memoria y la VM no dispone de tantos recursos, hace fallar a todo el sistema. Kafka deja de ser capaz de escribir por falta de memoria y el sistema entero termina por caer. (Se probaron con diversas modificaciones de memoria en driver y workers sin resultado exitoso).

- Caso 4. Spark en modo Standalone: Puesto que consume menos recursos, el sistema global es capaz de funcionar sin ninguna problemática.

Resultados

Pantallas de visualización



Current running cars	Current engine troubles	Current most selected car model
21	2	Volkswagen Passat

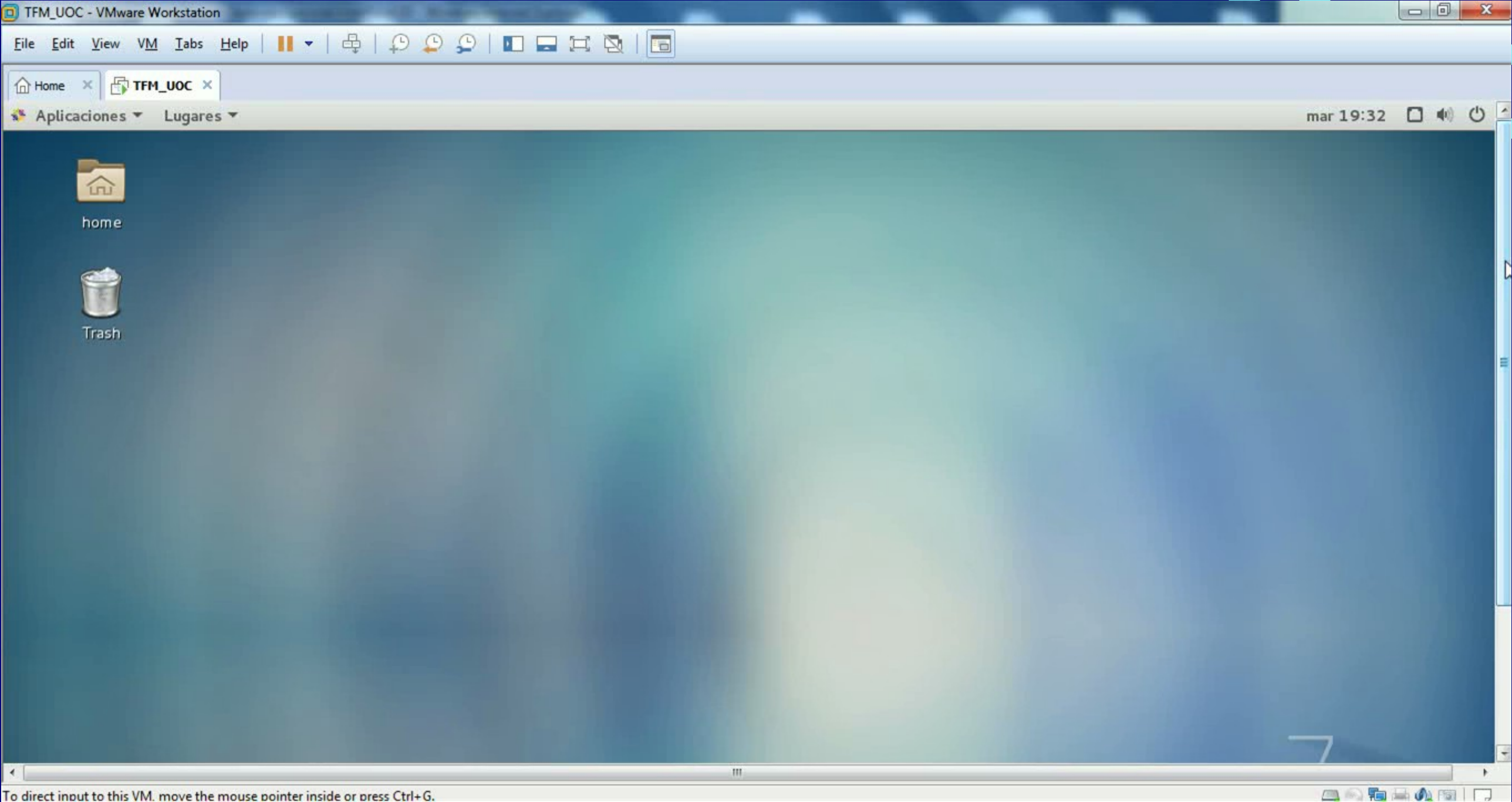
Por tal de mostrar los resultados generados por Spark de una forma más visual, se ha generado una pantalla de visualización con algunos de los agregados de negocio.

Demo de proyecto

El video muestra un ejemplo del funcionamiento del sistema completo.
En caso de no poder reproducir el video de Demo de la siguiente diapositiva, el video se encuentra disponible en las siguientes URL's:

Dropbox: https://www.dropbox.com/s/0m0rl8zakf2qs2r/demo_presentacion_tfm.avi?dl=0

Drive: https://drive.google.com/open?id=0B7yDD-M__K3uS2xrZUNfa205TTg


TFM_UOC - VMware Workstation
File Edit View VM Tabs Help
Home
TFM_UOC
Aplicaciones
Lugares
mar 19:32
home
Trash
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Resultados

Conclusiones

Este proyecto ha supuesto un reto bastante importante devido a la gran cantidad de tecnologías Big Data y de virtualización que toman parte en el.

Dentro de todos los conocimientos y objetivos del proyecto creo que se han conseguido todos los objetivos que se marcaron en el inicio. Ha quedado pendiente la ejecución en modo YARN de Spark debido a las deficiencias técnicas del equipo host del experimento.

Como evolutivos futuros, yo englobaría principalmente dos:

- El desarrollo de cada una de las tecnologías de forma individual. Debido a que están desplegadas sobre Docker, su despliegue en otros hardwares puede ser casi inmediata y eficiente.

- El remplazo de Spark Streaming en la parte final del sistema por otra tecnología de procesado en real time.