HADOOP

Extración, almacenamiento y análisis de TWeets

Juan Casado Ballesteros

GitHub: <https://github.com/JuanCasado/ADVANCED_DATABASES/tree/master/P3>

Índice

[Introducción 4](#_Toc40391038)

[Arquitectura lambda 4](#_Toc40391039)

[Capa batch 4](#_Toc40391040)

[Capa de servicio 4](#_Toc40391041)

[Capa en tiempo real 5](#_Toc40391042)

[Capa de acceso 5](#_Toc40391043)

[Trabajo realizado 6](#_Toc40391044)

[HADOOP 7](#_Toc40391045)

[Distribuciones 8](#_Toc40391046)

[Arquitectura 9](#_Toc40391047)

[Nuevos retos 10](#_Toc40391048)

[Hadoop SAS 10](#_Toc40391049)

[Escalado automático 10](#_Toc40391050)

[Flume 11](#_Toc40391051)

[Instalación 11](#_Toc40391052)

[HDFS 13](#_Toc40391053)

[PIG 14](#_Toc40391054)

[Mapreduce 14](#_Toc40391055)

[HBASE 15](#_Toc40391056)

[Thirf 15](#_Toc40391057)

[GRPC 15](#_Toc40391058)

[HIVE 16](#_Toc40391059)

[Despliegue 17](#_Toc40391060)

[Análisis de los datos 18](#_Toc40391061)

# Introducción

Se desea construir una arquitectura flexible capaz de analizar grandes cantidades de datos. La característica principal de estos datos es que son producidos de forma constante y en cantidades masivas por lo que una arquitectura tradicional de almacenamiento no sería capaz de manejarlos.

## Arquitectura lambda

Esto nos obliga a tener que adoptar una arquitectura propia del Big Data, la arquitectura Lambda que será implementada por medio de componentes del entorno Hadoop.

### Capa batch

Se utilizará Twitter como fuente de los datos. No obstante, tal y como se explicará esta fuente podría ser cambiada por otra de forma sencilla. Se utilizará Flume para realizar la extracción de los datos los cuales serán almacenados temporalmente sobre hdfs.

Las características de la capa batch es que los datos son procesados por lotes. Al utilizar hdfs como sistema de almacenamiento obtendremos ventajas adicionales como escalado, robustez y tolerancia a fallos.

### Capa de servicio

Los datos recolectados en bloques en la capa batch deben ser almacenados también en lotes en una base de datos que permita realizar consultas de forma rápida y eficaz. Las lecturas deben poder hacerse de forma aleatoria sobre esta base de datos, a diferencia de las escrituras que son en lotes.

Se utilizará Hbase, una base de datos columnar que se monta encima de hdfs. Esta base de datos hereda el escalado y la robustez de hdfs, permite lecturas aleatorias y escrituras en lotes y es sufrientemente flexible como para soportar variaciones en la estructura interna de los datos almacenados en ella.

Para transferir los datos de la capa batch a la capa de servicio se utilizará Pig. Esta herramienta utiliza la arquitectura de procesamiento mapreduce que se integra con hdfs por medio de Yarn para leer los datos almacenados en hdfs, procesarlos aplicando filtros sobre ellos y almacenándolos en Hbase.

Una vez que los datos han sido leídos de hdfs y almacenados en Hbase pueden ser borrados de hdfs.

Para realizar la práctica se ha utilizado Pig, pero otras arquitecturas de filtrado y procesamiento del entorno Hadoop podrían haberse utilizado en su lugar como Casacade también basada en mapreduce o Spark que propone su propia arquitectura de procesamiento.

### Capa en tiempo real

El objetivo de la capa en tiempo real dentro de la arquitectura Lambda es proporcionar a la capa de acceso los datos que se están almacenando o procesando en las capas batch o de servicio. Mientras que el procesamiento en lotes sucede los datos dentro del lote todavía no están disponibles de modo que es necesario una capa incremental que los proporcione durante el periodo en el que el batch se está procesando.

Dentro del ecosistema Hadoop hay múltiples opciones para implementar esta capa, la más habitual es utilizar Kafka, un sistema distribuido de paso de mensajes que admite subscripción. Si esta capa se utilizara debería haber un sink de Flume que la populara de datos incrementalmente.

### Capa de acceso

Acceder a los datos almacenados en Hbase no es la mejor solución dependiendo del uso que se vaya a dar a los datos. Para un acceso aleatorio a los datos como el que haría una aplicación Hbase puede ser una buena opción. No obstante, para analizar los datos, existen otras opciones mejores como Hive que proporcionan una interfaz similar a SQL integrada con mapreduce y Hbase para hacerlo.

# Trabajo realizado

Se ha implementado una versión de la arquitectura Lambda en la que se ha eliminado la capa en tiempo real. Debido al uso que se dará a los tweets y a la forma en la que serán analizados esta no era necesaria. Si se hubiera requerido de proporcionar análisis en tiempo real sobre los tweets capturados esta capa hubiera tenido que ser implementada. No obstante, debido a que lo que se pretende es primero capturar los tweets y luego analizarlos no existe esta necesidad.

Los retos que abordar son extraer la gran cantidad de tweets que se generan de forma constante, filtrarlos y procesarlos de modo que nos quedemos solo con aquellos que sean de utilidad, almacenarlos con una estructura que nos permita analizarlos convenientemente y proporcionar un método rápido de acceso a toda esa información recolectada.

La instalación de los distintos componentes de Hadoop que forman la arquitectura Lambda mencionada se ha realizado sobre contenedores. Hacer esto permite un despliegue automatizado y escalable. Otras ventajas de este despliegue es la reducida dependencia entre los componentes utilizados. Cada uno de los bloques que forma parte de la arquitectura depende de los otros tan solo en las interfaces web que expone y no es la estructura de los directorios, de la instalación realizada o de las versiones de los componentes instalados.

La versión de todos los componentes utilizados es la última versión estable disponible: Hadoop-3.2.1, Hbase-2.2.4, Pig-0.17.0, Hive-3.1.2, Flume-1.7.0 y Zookeeper-3.4.10. Algunos de estos componentes no son plenamente compatibles con el resto debido a incompatibilidades en las librerías que utilizan o diferencias de configuración. No obstante, debido a que comparten una interfaz de comunicación estable y estandarizada se les ha podido hacer convivir gracias a su despliegue en contenedores.

# HADOOP

La cantidad de datos que se producen diariamente crece de forma constante, los recursos necesarios para poderlos analizar son cada vez mayores y las arquitecturas de almacenamiento y procesamiento tradicionales no son capaces seguir este ritmo.

Son muchas las soluciones propuestas para solventar estos nuevos problemas. Algunas de estas soluciones se centran en resolver problemas concretos como almacenamiento distribuido (Big table, Cassandra, Ceph), acceso uniforme a los datos (Big query, Presto) o la computación distribuida y escalable (Kubernetes, Disco). Cada una de estas soluciones se adapta a resolver problemas concretos y lo hacen desde filosofías muy distintas, pero todas tien tres principios en común.

* **Escalado horizontal**: el escalado horizontal consiste en utilizar comodity hardware, es decir, equipos tradicionales que no tienen por qué pertenecer al mainframe, para aumentar la capacidad de cómputo. En lugar de utilizar un equipo más grande, caro y potente, se logra mayor potencia de cálculo por medio de la agregación de más equipos tradicionales.
* **Resistencia ante fallos**: cuantos más equipos se estén utilizando mayores serán las posibilidades de que uno falle. Si un equipo falla se debe evitar a toda costa perder información utilizando técnicas de replicación y de aseguramiento de la coherencia de los datos a lo largo de los equipos involucrados en almacenarla y procesarla.
* **Alto rendimiento**: un problema habitual en las arquitecturas tradicionales es que la capacidad de almacenamiento y la potencia de cómputo no crece linealmente con la cantidad de equipos pertenecientes al sistema. En las arquitecturas Big Data se pone especial cuidado en que añadir un equipo adicional al sistema suponga un aumento lineal en el rendimiento. De otro modo se entrarían ante un cuello de botella que limitaría su capacidad de procesar o almacenar datos alcanzada cierta cantidad de estos.

Hadoop es una solución más a todos estos problemas. Es un ecosistema rico y variado formado por múltiples aplicaciones que se integran a distintos niveles. Algunas de las características que permiten distinguir las aplicaciones que pertenecen o no al entorno Hadoop es que son Open Source y mantenidas por la plataforma Apache.

No obstante, y debido al gran crecimiento de Hadoop en los últimos años la línea entre lo que es Hadoop y lo que no es cada vez más difusa. Son más y más las aplicaciones que se integran en el ecosistema, algunas aplicaciones que originalmente formaron parte de él están empezando a quedar desactualizadas. Mientras que otras nunca quedaron claro si formaban parte o no como es el caso de Cassandra que a pesar de ser desarrollado por Apache según la fuente es considerado o no parte de Hadoop, pues no se basa en hdfs.

## Distribuciones

Hadoop puede instalarse de forma nativa sobre Windows o Linux, en una máquina virtual o sobre contenedores. En todos esos casos se corre el gran riesgo de tener problemas con la instalación o problemas de compatibilidad entre sus componentes.

Esto se debe a que en ocasiones los productos Open Source pueden divergir en las interfaces por las que se comunican, en las tecnologías que utilizan o simplemente no ser actualizados de forma homogénea.

Para suplir estas carencias múltiples empresas como Cloudera y Hortonworks, recientemente fusionadas o mapR sirven distribuciones de Hadoop con diversos componentes ya instalados y listos para usarse con garantías de que funcionarán adecuadamente. Las ventajas de utilizar una de estas distribuciones se ve ampliada con soporte técnico, cursos y certificaciones para empleados, tutoriales, ejemplos y blogs donde poder preguntar dudas.

Las distribuciones de Hadoop también proporcionan componentes adicionales que enriquecen el entorno como entornos gráficos desde los que poder configurar y extender la instalación, visualizar su estado, así como componentes para extenderla como la base de datos Impala perteneciente al entorno de Cloudera.

## Arquitectura

Se elijan los componentes que se elijan dentro del entorno de Hadoop existe una arquitectura base sobre la que estos puede ser instalados. Esta arquitectura tiene en su base el sistema de almacenamiento distribuido hdfs. Este sistema de almacenamiento proporciona la robustez y la tolerancia a fallos de Hadoop a lo largo de cualquier cantidad de nodos.

Las aplicaciones pueden instalarse directamente sobre hdfs sin necesitar componentes adicionales como es el caso de Hbase o Flume.

Por encima de hdfs se encuentra Yarn que es un gestor de recursos. Su misión es proporcionar una interfaz del sistema de almacenamiento suficientemente general como para instalar entornos de cómputo distribuido. Algunos de estos entornos son Spark, Mapreduce o Tez.

Mapreduce es el entorno de cómputo original de Hadoop, de echo en un origen este se montaba directamente sobre hdfs sin posibilidad de utilizar otros entornos de cómputo. En la actualidad Mapreduce está perdiendo popularidad frente a Tez que ya es recomendado frente a Mapreduce en múltiples aplicaciones como Hive o Pig, por el contrario, Spark proporciona su propio mini entorno por encima de Hadoop no sin ello perder la compatibilidad con esas mismas aplicaciones gracias a la interfaz de Yarn desde la que se pueden comunicar.

Al margen del sistema de almacenamiento y del entorno de cómputo dentro de Hadoop han surgido múltiples aplicaciones en las que concentrar servicios como es el caso de Zookeeper el cual es un gestor distribuido para almacenar nombres y configuraciones con la intención de sincronizar servicios distribuídos. Zookeeper por ejemplo implementa el servicio de sincronización entre el master activo de Hbase y sus regiones. Otro ejemplo bajo esta misma filosofía es Kafka que proporciona un context broker basado en streams distribuídos y en paso de mensajes que admite subscripciones con notificación en tiempo real.

Estas aplicaciones como Kafka o Zookeeper son también utilizadas en entornos fuera del ecosistema de Hadoop.

## Nuevos retos

Hadoop aunque es una gran solución dentro del entorno del Big Data presenta dos grandes debilidades o carencias, ambas notablemente experimentadas a lo largo de la realización de esta práctica.

### Hadoop SAS

Muchas bases de datos como Firebase, Big Table o MongoDB ofrecen servicios SAS para acceder a ellas. Con esto se logran evitar problemas de instalación y de mantenimiento. La mayoría de los componentes de Hadoop como Hbase, Hive o Kafka podrían seguir modelos similares. Resulta extraño ver como otros productos similares, son comercializados de esta forma mientras que Hadoop no lo es.

### Escalado automático

Algunos de los componentes de Hadoop admiten escalado dinámico como es el caso de los nodos de almacenamiento y los nodos de procesamiento en hdfs. Otras partes de Hadoop por el contrario solo admiten escalado estático como es el caso de las regiones de Habse o el quorum de Zookeeper.

Actualmente servicios como las rolling updates o el escalado automático son una parte esencial de los sistemas de orquestación como Kubernetes o Mesos. Estos servicios son la base de los centros de computación cloud pues permiten a las bases de datos y a los nodos de procesamiento adaptarse de forma dinámica y en tiempo real a las cargas de trabajo a las que sean cometidos.

# Flume

Flume es una herramienta de ingestión de datos. Su misión es tomar datos de una fuente concreta y dejarlos en otra. Flume es compatible con gran cantidad de fuentes de datos como las APIs que las redes sociales ponen a disposición de los usuarios, bases de datos o archivos almacenados en hdfs u otros sistemas de almacenamiento.

En nuestro caso utilizaremos como fuente de datos la API de Twiter pero modificando la configuración de flume se podrían obtener datos de cualquiera de las otras posibles fuentes mencionadas. Posteriormente esos datos serán depositados en hdfs. Podrían depositarse en otros destinos como Kafka o directamente sobre Hbase.

## Instalación

Para extraer los datos de Twitter se utilizará un programa escrito en Java cuya misión es permitir aplicar filtros sobre la fuente de Flume. Estos filtros son opcionales y pueden restringir la ingestión de datos dentro de un área geográfica o haciendo que contengan solo ciertas palabras. Para que Flume pueda utilizar esta fuente de datos se ha compilado el código junto con la librería Twitter4j en un .jar que se ha dejado junto al resto de librerías con las que Flume ya viene equipado.

Como salida de los datos se ha utilizado hdfs y como canal se ha utilizado MemChanel. Con esto quedan cubiertos los tres componentes de los que se compone un agente de Flume: fuente, canal y salida.

Esta configuración del agente se escribe en un archivo .conf cuya única peculiaridad es que ya que el namenode de hdfs es remoto se debe indicar su puerto junto a la dirección de hdfs, en nuestro caso hdfs://namenode:9000/ruta/donde/dejar/los/archivos.

Las dependencias internas de Flume consisten en hadoop-client que debe ser el mismo que el que esté instalado en el namenode y de Zookeeper-3.5.2. La versión de Zookeeper 3.4.10 que es la expuesta externamente es compatible con el uso que hace Flume con la 3.5.2.

Con respecto al uso de red Flume no necesita ser accesible desde el exterior, no obstante, requiere que el namenode en sus dos interfaces, el nodemanager y al menos un Zookeeper y un datanode estén disponibles.

Finalmente, el agente de Flume puede ser lanzado con:

**flume-ng agent –conf conf -f /archivo.conf -n NombreDelAgente**

Algunas limitaciones de utilizar Flume para esta tarea es que la API de Twitter en su modo de uso gratuito impone limitaciones en la cantidad de agentes disponibles con una misma acreditación y en el tiempo en el que el agente puede estar recolectando los datos. De modo que solo podrá haber un agente activo extrayendo datos de la API y este sufrirá un error cuando lleve cierto tiempo en ejecución.

Se puede ver desde el explorador de ficheros web de Hadoop como Flume escribe los datos que recolecta en distintos ficheros de modo que la recolección sea lo más eficiente posible:

A screenshot of a cell phone

Description automatically generated

# HDFS

# PIG

## Mapreduce

# HBASE

## Thirf

## GRPC

# HIVE

# Despliegue

# Análisis de los datos