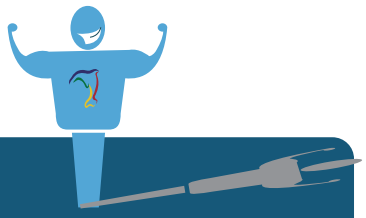
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

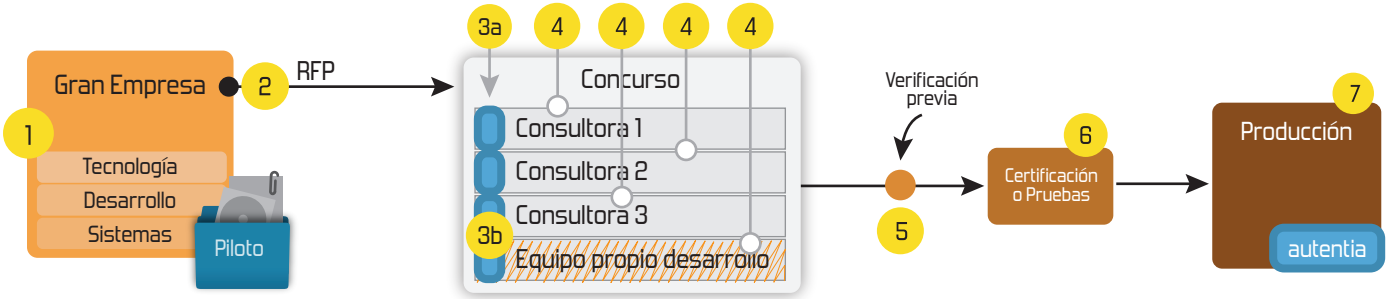
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Francisco Javier Martínez Páez

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero Técnico en Telecomunicaciones

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2012-02-15 Tutorial visitado 3 veces Descargar en PDF

Apache Hadoop-HDFS

Este tutorial es una continuación de este otro. Os recomiendo que antes de este tutorial visitéis este otro, ya que van encadenados.
Los fuentes son los mismos que el otro tutorial, por lo que no os pongo el enlace.

Introducción

Todo sistema de ficheros distribuido tiene un "objetivo" principal: solucionar el problema de almacenar la información que supera las capacidades de una única máquina. Para superar este problema, un sistema de ficheros distribuido gestionará y permitirá el almacenamiento de la información en diferentes máquinas conectadas a través de una red, haciendo transparente al usuario la complejidad interna de su gestión.

HDFS es un sistema de ficheros pensado para el almacenamiento de ficheros "grandes" (por encima de 100 MB) y en la que el acceso a esa información está orientado hacia procesamiento en batch o lectura de tipo "write once"-"read-many-times" (ideal para temas de MapReduce, pero no para necesidades de baja latencia) y cuyo diseño está pensado para ser ejecutado en máquinas "baratas".

En un cluster de HDFS encontramos dos tipos de nodos diferentes:

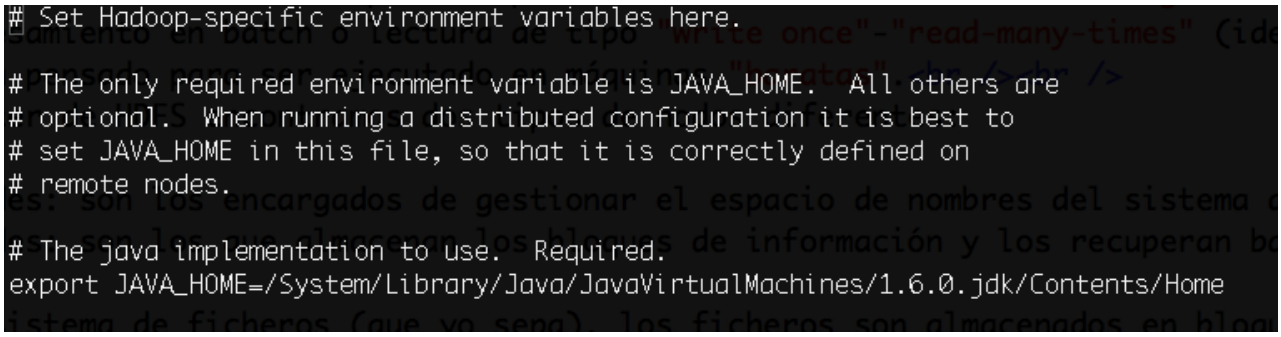
- Namenodes: son los encargados de gestionar el espacio de nombres del sistema de ficheros
- Datanodes: son los que almacenan los bloques de información y los recuperan bajo demanda

Como todo sistema de ficheros (que yo sepa), los ficheros son almacenados en bloques de un mismo tamaño (block size), que en un sistema de ficheros tradicional suele ser de unos 4KB. En HDFS también existe este concepto, pero el tamaño del bloque es mucho mayor (64MB por defecto) para minimizar el coste de acceso a los bloques, y lógicamente, los bloques de un mismo fichero no tienen porque residir en el mismo nodo.

Configuración de Apache Hadoop Distributed File System

Lo primero que haremos será configurar el sistema de ficheros distribuido de Hadoop, en nuestro caso usaremos el modo pseudo-distribuido que nos permite simular un cluster en nuestra máquina local.
Los ficheros de configuración de hadoop se encuentra en el directorio "conf" incluido en la distribución de Hadoop.

El primer fichero que vamos a editar es hadoop-env.sh. En este fichero configuraremos JAVA_HOME (usad la vuestra):



```
# Set Hadoop-specific environment variables here.

# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.  Required.
export JAVA_HOME=/System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home
```

Después editaremos el fichero core-site.xml:

view plain print ?

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost/</value>
    </property>
</configuration>
```

La propiedad *fs.default.name* nos permite indicar el sistema de ficheros por defecto de nuestra distribución de Hadoop. En este caso hemos especificado que el sistema de ficheros por defecto será "hdfs", que el "namenode" será localhost y el puerto 8020 (puerto por defecto).

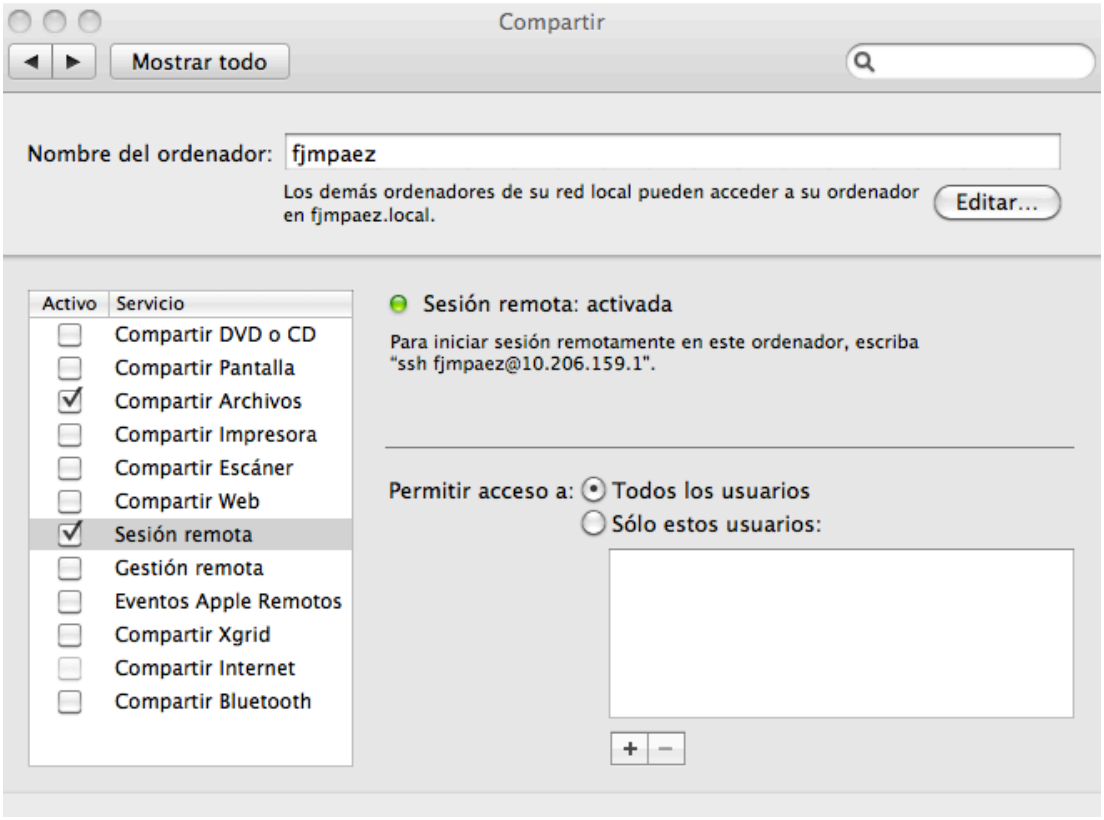
A continuación editaremos el fichero hdfs-site.xml:

view plain print ?

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

Esta propiedad indica el nivel de replicación de cada bloque (tolerancia a fallos), que por defecto es 3. En modo pseudo-distribuido en el que sólo existe una máquina en el cluster no tiene sentido tener un valor superior a 1.

Ahora debemos configurar SSH en nuestra máquina, para permitir que Hadoop pueda inicializar los nodos del cluster. Además debemos configurarlo para permitir que se pueda conectar sin password. Por lo tanto, lo primero que debemos hacer es tener instalado ssh.
En mi caso (Mac)...debemos activar Sesión Remota en Preferencias del Sistema > Compartir:



Debemos permitir acceso sin login:

ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa_no.pub >> ~/.ssh/authorized_keys

Si tenéis claves ya generadas, haced backup de las claves, para dejárlo luego todo como estaba (estoy pensando por ejemplo en los que usáis github)

Para probar que lo hemos hecho bien:

ssh localhost (debemos poder entrar sin password)

Una vez hemos configurado todo lo necesario, debemos formatear el sistema de ficheros ejecutando:

hadoop namenode -format

Ahora sólo queda arrancar HDFS:

start-dfs.sh

Vamos a copiar algo a nuestro sistema de ficheros. Voy a copiar los ficheros que me descargué del tutorial anterior en un directorio llamado bible:

hadoop fs -mkdir hdfs://localhost/bible

hadoop fs -copyFromLocal /bible/ hdfs://localhost/bible*

No os preocupéis de los errores que os muestra (tiene que ver con que no puede replicar los bloques)

hadoop fs -ls hdfs://localhost/bible

```
fjmpaez:conf fjmpaez$ hadoop fs -ls /bible
Found 32 items
-rw-r--r--   1 fjmpaez supergroup      52963 2012-02-14 16:17 /bible/1corinth.webarchive
-rw-r--r--   1 fjmpaez supergroup      13459 2012-02-14 16:17 /bible/1john.webarchive
-rw-r--r--   1 fjmpaez supergroup      14165 2012-02-14 16:17 /bible/1peter.webarchive
-rw-r--r--   1 fjmpaez supergroup      10172 2012-02-14 16:17 /bible/1thess.webarchive
-rw-r--r--   1 fjmpaez supergroup      13335 2012-02-14 16:17 /bible/1timothy.webarchive
-rw-r--r--   1 fjmpaez supergroup      34040 2012-02-14 16:17 /bible/2corinth.webarchive
-rw-r--r--   1 fjmpaez supergroup       1646 2012-02-14 16:17 /bible/2john.webarchive
-rw-r--r--   1 fjmpaez supergroup       9112 2012-02-14 16:17 /bible/2peter.webarchive
-rw-r--r--   1 fjmpaez supergroup       5717 2012-02-14 16:17 /bible/2thess.webarchive
-rw-r--r--   1 fjmpaez supergroup       9666 2012-02-14 16:17 /bible/2timothy.webarchive
-rw-r--r--   1 fjmpaez supergroup       1692 2012-02-14 16:17 /bible/3john.webarchive
-rw-r--r--   1 fjmpaez supergroup     136319 2012-02-14 16:17 /bible/acts.webarchive
-rw-r--r--   1 fjmpaez supergroup      11264 2012-02-14 16:17 /bible/colossia.webarchive
-rw-r--r--   1 fjmpaez supergroup     152744 2012-02-14 16:17 /bible/deut.webarchive
-rw-r--r--   1 fjmpaez supergroup      17169 2012-02-14 16:17 /bible/ephesian.webarchive
-rw-r--r--   1 fjmpaez supergroup     177528 2012-02-14 16:17 /bible/exodus.webarchive
-rw-r--r--   1 fjmpaez supergroup      17380 2012-02-14 16:17 /bible/galatian.webarchive
-rw-r--r--   1 fjmpaez supergroup     207293 2012-02-14 16:17 /bible/genesis.webarchive
-rw-r--r--   1 fjmpaez supergroup      39353 2012-02-14 16:17 /bible/hebrews.webarchive
-rw-r--r--   1 fjmpaez supergroup      12853 2012-02-14 16:17 /bible/james.webarchive
-rw-r--r--   1 fjmpaez supergroup     103568 2012-02-14 16:17 /bible/john.webarchive
-rw-r--r--   1 fjmpaez supergroup       3693 2012-02-14 16:17 /bible/jude.webarchive
-rw-r--r--   1 fjmpaez supergroup     132548 2012-02-14 16:17 /bible/levit.webarchive
-rw-r--r--   1 fjmpaez supergroup     142054 2012-02-14 16:17 /bible/luke.webarchive
-rw-r--r--   1 fjmpaez supergroup      83355 2012-02-14 16:17 /bible/mark.webarchive
-rw-r--r--   1 fjmpaez supergroup     131405 2012-02-14 16:17 /bible/matthew.webarchive
-rw-r--r--   1 fjmpaez supergroup     184204 2012-02-14 16:17 /bible/numbers.webarchive
-rw-r--r--   1 fjmpaez supergroup       2469 2012-02-14 16:17 /bible/philemon.webarchive
-rw-r--r--   1 fjmpaez supergroup      12157 2012-02-14 16:17 /bible/philipp.webarchive
-rw-r--r--   1 fjmpaez supergroup      64953 2012-02-14 16:17 /bible/rev.webarchive
-rw-r--r--   1 fjmpaez supergroup      53723 2012-02-14 16:17 /bible/romans.webarchive
-rw-r--r--   1 fjmpaez supergroup       5384 2012-02-14 16:17 /bible/titus.webarchive
```

La información es similar al comando ls, con la distinción que en la segunda columna nos indica el factor de replicación del fichero.

Vamos a comprobar que recuperamos la misma información que guardamos (probamos a descargar uno de los ficheros y le hacemos un MD5):

hadoop fs -copyToLocal hdfs://localhost/bible/luke.webarchive /lucas.txt

md5 /bible/luke.webarchive /lucas.txt

```
MD5 (/Trabajo/tutoriales/hadoop/data/luke.webarchive) = 90159311b5fa9d98c7aee3b98be6355c
MD5 (/lucas.txt) = 90159311b5fa9d98c7aee3b98be6355c
```

Si no os fiáis del MD5, podéis comparar los ficheros (podríamos haber descargado un fichero distinto y encontrar un colisión en MD5, nunca se sabe)

MapReduce en cluster.

Cuando queremos ejecutar un MapReduce (un Job en jerga Hadoop) en cluster (o pseudo-cluster que a efectos prácticos es lo mismo), se desencadena una serie de acciones que podríamos resumir de esta manera:

1. Se lanza un "Job" desde una aplicación cliente
2. JobClient solicita al JobTracker un identificador de Job
3. JobClient procesa los datos de entrada y los divide en piezas de tamaño fijo ("input splits")
4. JobClient copia los recursos necesarios para la ejecución del programa al sistema de ficheros (HDFS) incluído el "jar" que contiene las clases a ejecutar y los "input splits"
5. JobClient notifica a JobTracker que el "job" está preparado para ejecutarse
6. JobTracker encola el "job"
7. El Job Scheduler recoge el "job" y lo inicializa. Para ello recoge los "input splits" procesados y crea una "Map Task" por cada "split". El número de "Reduce Task" viene dado por el valor de la propiedad "mapred.reduce.tasks". A cada tarea se le asigna un identificador
8. Los TaskTrackers se "enbuclan" notificando periódicamente su estado al JobTracker (heartbeat) y su disponibilidad para correr tareas. Cuando un TaskTracker está disponible, el JobTracker le asigna una tarea para correrla

9. Para correr una tarea, el TaskTracker localiza el jar y los ficheros necesarios para la ejecución en el sistema distribuido de ficheros, los copia en local, descomprime el jar y crea un TaskRunner. Cada TaskRunner lanza su propia JVM, y comunica al TraskTracker el progreso de la tarea periódicamente
10. Cuando el JobTracker es notificado de que la última tarea ha sido completada, cambia el estado del trabajo a "successful", notifica su finalización (si se ha configurado para ello en "job.end.notification.url") y libera los recursos utilizados.

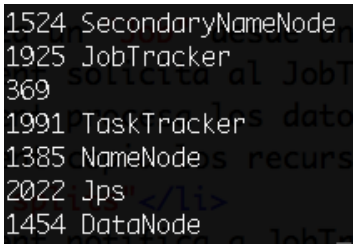
Vamos ahora a ejecutar el mismo ejemplo de MapReduce que preparamos para el tutorial anterior. Lo primero que debemos hacer es configurar el JobTracker (Gestiona o coordina las ejecuciones de MapReduce en el cluster). Para ello editaremos el fichero mapred-site.xml:

```
view plain  print  ?
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
<property>
    <name>mapred.job.tracker</name>
    <value>localhost:8021</value>
</property>
</configuration>
```

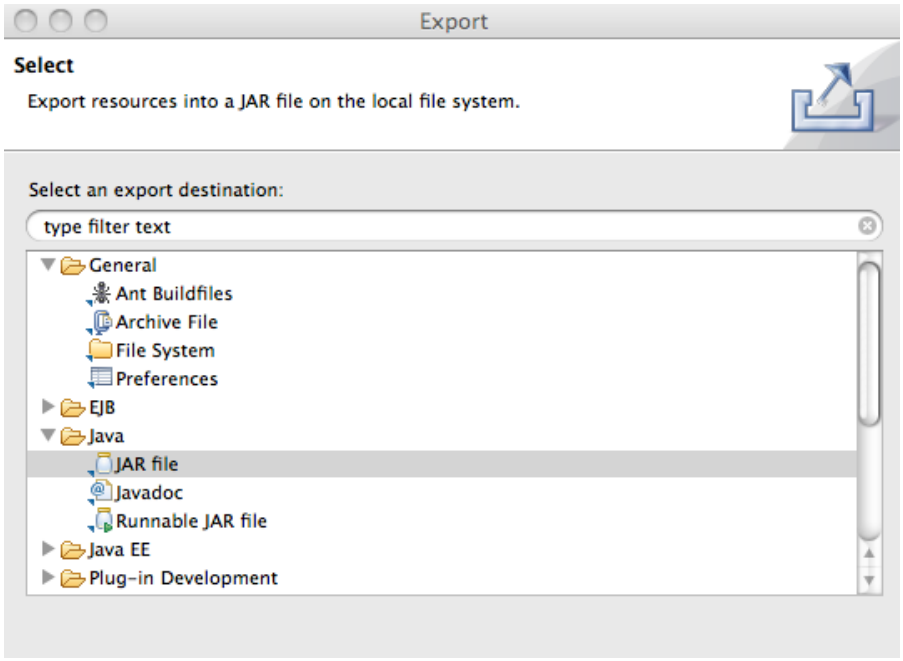
Una vez configurado, estamos preparados para "arrancar" el JobTracker:

start-mapred.sh

Para comprobar que tenemos todo arrancado correctamente, podemos ejecutar jps para ver los procesos java arrancados: *jps*



El código fuente es el mismo que el usado el anterior, pero para poder ejecutar el programa necesitamos generar el "jar", ya que este debe ser distribuido en HDFS. Para generar el jar lo podéis hacer desde el Eclipse, en "Export":



Para lanzar el programa, ejecutad: (las rutas variarán en función de vuestro entorno)

hadoop jar /bible/bibleWordCounter.jar com.autentia.tutoriales.hadoop.BibleWordCounter hdfs://localhost/bible file:///bible/resultado

Comprobaremos el resultado, para ver si coincide con el del tutorial anterior (y es así):

```
CHRIST  525
CHRIST'S    16
CHRISTIAN   2
CHRISTIANS  1
CHRISTS 2
CHRYSOLITE  1
CHRYSOPRASUS    1
CHURCH  70
CHURCHES    35
```

Si habéis hecho los dos tutoriales comprobaréis que esta ejecución ha sido como 8 veces más lenta (de 10 segundos a 80 segundos), y alguno dirá que vaya "mejora". A lo que yo respondería que *para este viaje no hacen falta alforjas*, ya que la cantidad de información a analizar no es lo suficientemente grande como para merecer la pena distribuirla, y por otro lado, no debemos perder de vista que no estamos usando un cluster real sino una sóla máquina (pseudo-distribuido).

Conclusiones.

Espero que estos dos tutoriales os hayan servido para "comprender" qué es Apache Hadoop MapReduce y HDFS, y sobre todo, dónde o cuándo tiene sentido usar su "potencia".
Por lo tanto, cuando tengáis necesidad de analizar grandes cantidades de información y obtener respuestas en "poco" tiempo, creo que es el momento de que os planteéis usar Hadoop, y si no os atrevéis sólos, pues ya sabéis, Autentia está ahí para ayudaros.