

openTSDB

Francisco Rodríguez Redondo

19 de enero de 2018

https://bigdatafran.github.io/big_data/

Índice general

1. Primeros pasos.	5
1.1. Introducción a OpenTSDB.	5
1.2. Instalación de OpenTSDB.	7
1.2.1. Instalación directa sobre el sistema operativo CentOS.	7
1.2.1.1. Configuración de OpenTSDB con HBase.	11
1.2.2. Instalación utilizando Ambari.	16
1.3. Primer ejemplo con OpenTSDB.	16
1.4. Configuración de OpenTSDB.	17
2. Estructura de OpenTSDB.	22
2.1. Agregación.	22
2.2. Downsampling.	22
2.3. Rate (o Razón).	23
2.4. Filtros.	24
2.5. Rollup's y Pre-aggregates.	24
2.6. Componentes de una query.	25
2.7. UUIDs y TSUIDs.	25
2.7.1. UUID.	25
2.7.2. TSUID.	26
2.8. Metadatos.	26
2.8.1. UIDMeta.	26
2.8.2. TSMeta.	26
2.8.3. Annotations.	26
2.9. Fechas y Tiempos en OpenTSDB.	26
2.9.1. Fechas y tiempos relativos.	27
2.9.2. Fechas y tiempos absolutos.	27
3. Trabajando con los datos de OpenTSDB.	29
3.1. Herramientas CLI.	29
3.1.1. uid.	31
3.1.1.1. grep.	31
3.1.1.2. assign.	32
3.1.1.3. rename.	32
3.1.1.4. delete.	32
3.1.1.5. fsck.	33
3.1.1.6. metasync.	33
3.1.1.7. metapurge.	33

Índice general

3.1.1.8.	treesync.	34
3.1.1.9.	treepurge.	34
3.1.2.	mkmetric.	34
3.1.3.	import.	34
3.1.4.	query.	35
3.1.5.	fsck.	36
3.1.6.	scan.	36
3.1.7.	search.	36
3.1.8.	tsd.	36
3.2.	Telnet API	36
3.2.1.	put.	37
3.2.2.	rollup.	37
3.2.3.	histogram.	37
3.2.4.	stats.	37
3.2.5.	version.	37
3.2.6.	help.	37
3.2.7.	dropcaches.	38
3.2.8.	diediedie.	38
3.3.	HTTP API	38
3.3.1.	Filtros Query. ¹	39
3.3.1.1.	Agrupación de datos.	39
3.3.1.2.	Formato Query de una métrica.	40
3.3.1.3.	Formato Query de un TSUID.	41
3.3.2.	API Endpoints.	41
3.3.2.1.	/s	41
3.3.2.2.	/api/aggregators.	41
3.3.2.3.	/api/annotation.	42
3.3.2.4.	/api/annotation/bulk.	43
3.3.2.5.	/api/config.	43
3.3.2.6.	/api/config/filters.	43
3.3.2.7.	/api/dropcaches.	44
3.3.2.8.	/api/put.	44
3.3.2.9.	/api/rollup	47
3.3.2.10.	/api/histogram	47
3.3.2.11.	/api/query.	47
3.3.2.12.	/api/search.	50
3.3.2.13.	/api/serializers.	50
3.3.2.14.	/api/stats.	51
3.3.2.15.	/api/suggest.	51
3.3.2.16.	/api/tree.	52
3.3.2.17.	/api/uid/assign.	52

¹Para ver los filtros disponibles en OpenTSDB 2.2 en adelante, se puede utilizar en la barra de direcciones del navegador la expresión <http://localhost:8091/api/config/filters>

Índice general

3.3.2.18. /api/uid/tsmeta.	53
3.3.2.19. /api/uid/uidmeta.	55
3.3.2.20. /api/version	56
4. Utilidades.	57
4.1. tcollector.	57
4.1.1. Instalación de tcollector.	57
4.2. Potsdb.	58
4.3. Cliente R.	59
4.4. Clientes con Java.	60
4.4.1. Opentsdb-java-client.	60
4.4.2. Construir un cliente vía la clase HttpURLConnection.	63
A. Código fichero google_intraday.py.	66
B. Google Finance.	70

https://bigdatafran.github.io/big_data/

1. Primeros pasos.

La tecnología mapreduce que descansa sobre Hadoop ha permitido desarrollar muchas herramientas dentro el mundo «BigData» para el tratamiento de datos masivos. Muchas de estas tecnología se engloban dentro del denominado ecosistema de Hadoop que facilitan mucho la labor dentro del apartado del tratamiento masivo de la información.

En el presente documento se va a desarrollar un tecnología de tratamiento de series temporales, que aunque no se encuadra por completo dentro del ecosistema de Hadoop, sí utiliza su tecnología para el tratamiento masivo de grandes cantidades de información. En concreto, se va a proceder a desarrollar y exponer el sistema openTSDB, que permite trabajar con series temporales soportadas bajo el sistema de ficheros HDFS (Hadoop Distributed File System) que presenta Hadoop.

En este documento se pretende ofrecer todo el potencial que presenta openTSDB, y para ello se partirá exponiendo dos método para su implementación, para posteriormente exponer diferentes posibilidades que ofrece esta base de datos para poder por ejemplo cargar o leer datos de la misma.

Para centrar el tema, hay que tener en cuenta que la instalación se ha realizado sobre la distribución de Hadoop facilitada por HortonWorks, y que para su instalación se ha empleado la máquina virtual Oracle VirtualBox. Las características de los medios utilizados han sido las siguientes:

- Sistema operativo: CentOS release 6.9 (Final).
- Versión Hadoop: 2.7.3
- Versión HortonWorks: 2.6

1.1. Introducción a OpenTSDB.

La base sobre la que se sustenta OpenTSDB es un «Time Series Daemon» (TSD) que permite ofrecer toda la información que presentan el sistema. La interacción con este sistema se realiza mediante estos TSD, de tal manera que cada uno de los TSD con los que se trabaja en un momento determinado son independientes entre ellos, además hay que tener en cuenta que cada uno de estos TSD, trabajan contra la base de datos HBASE del ecosistema de Hadoop, por lo que es imprescindible tener levantada esta base. Existen tres métodos para poderse comunicar el usuario con los TSD:

1. El protocolo telnet.
2. Mediante el API HTML

1. Primeros pasos.

3. Mediante unos sencillos comandos diseñados al efecto y que se comentarán en futuros apartados.

Un esquema gráfico de todo este sistema lo podemos ver en la figura 1.1 obtenida de la [página web que da soporte a OpenTSDB](#).

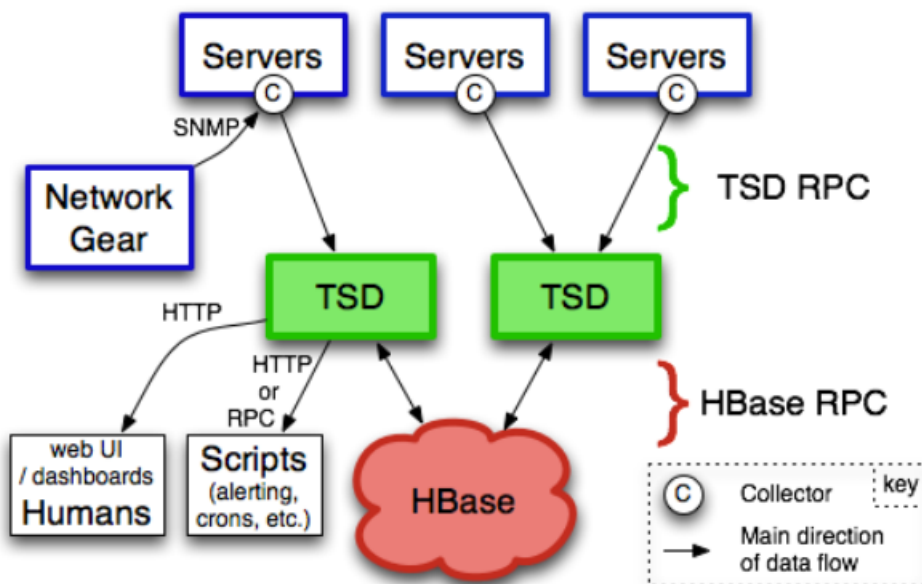


Figura 1.1.: Esquema OpenTSDB

En OpenTSDB cada registro de la base de datos se denomina «data point»¹ y la estructura del mismo se muestra en la figura 1.2 y a continuación se detalla esa estructura:

- La primera parte sería el nombre de la métrica que se refleja en el registro.
- La segunda parte hace referencia al momento en el que se hace la medición. Este tiempo tiene una estructura UNIX y representa el tiempo en segundos o mili segundos desde el 1 de enero de 1970.
- A continuación vendría el valor tomado.
- Después vienen los «tags» o etiquetas que no son más que pares del tipo clave-valor

De esta manera se podrán construir diferentes series temporales con los data point's que figuran en la base de datos. Así por ejemplo, en el ejemplo mostrado en la figura 1.2 se podría construir una serie temporal que contenga sólo los data point's correspondientes a la métrica «mysql.bytes_received» que estaría construida por tres registros. Otra serie

¹En castellano la denominación sería punto de datos, pero se ha preferido mantener la denominación anglosajona por entender que refleja mejor el hecho que pretende representar

1. Primeros pasos.

Nombre métrica	Tiempo	Valor	Tag_1	Tag_2
mysql.bytes_received	1287333217	327810227706	schema=foo	host=db1
mysql.bytes_sent	1287333217	6604859181710	schema=foo	host=db1
mysql.bytes_received	1287333232	327812421706	schema=foo	host=db1
mysql.bytes_sent	1287333232	6604901075387	schema=foo	host=db1
mysql.bytes_received	1287333321	340899533915	schema=foo	host=db2
mysql.bytes_sent	1287333321	5506469130707	schema=foo	host=db2

Figura 1.2.: Estructura de los datos

temporal se podría construir para la métrica «mysql.bytes_sent» que tendría otros tres registros.

Pero a mayores, también se pueden construir otras series temporales utilizando también los tags que existen para cada point, y de esta manera se podrán seleccionar los registros con un nombre de la métrica igual a «mysql.bytes_received» y que además tenga en un tag el par de valores schema=foo. De esta manera y dentro de una misma base de datos, las series temporales que se pueden construir pueden alcanzar un número elevado y directamente proporcional al número de métricas y «tags» que contenga. Por lo tanto, se obtendrá una serie temporal por cada combinación que se haga de una métrica y uno o varios tags que tiene cada data point. Según se recomienda en página web que desarrolla esta tecnología, es aconsejable que no tener más de 6-7 tags por cada data point.

OpenTSDB ofrece un diseño propio para poder ver los datos de cada serie temporal que se quiera seleccionar, y en este sentido ofrece una interfaz gráfica donde se van a poder representar los datos que se deseen. En apartados posteriores se mostrará esta herramienta de indudable valor para el usuario de esta tecnología.

1.2. Instalación de OpenTSDB.

En el siguiente apartado se va a mostrar, dos formatos de instalación de OpenTSDB. Por un lado se procederá a su instalación de una forma directa sobre el sistema operativo Centos que ofrece VirtualBox, y por otro lado se utilizarán las facilidades que ofrece Ambari, como gestor de servicios de HortonWorks. En ambos casos es muy necesario tener en cuenta, que se precisa tener en marcha el sistema HBASE del ecosistema Hadoop.

1.2.1. Instalación directa sobre el sistema operativo CentOS.

Ya se ha dicho anteriormente que hay que tener la precaución de comprobar que HBASE está en funcionamiento pues sobre este sistema descansa OpenTSDB. Pero además también hay que comprobar que zookeeper también está accesible, para comprobarlo un método puede consistir en utilizar el comando «telnet» de linux, junto con la ejecución del comando «stats». La ejecución de estos comandos se muestra en la figura 1.3, por lo tanto obteniendo estos resultados se puede tener la seguridad de que Zookeeper está accesible y por lo tanto se verifica esa condición para seguir hacia adelante con la instalación

1. Primeros pasos.

```
[root@sandbox ~]# telnet localhost 2181
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^'.
stats
Zookeeper version: 3.4.6-129--1, built on 05/31/2017 03:01 GMT
Clients:
 /172.17.0.2:60540[1] (queued=0,recved=3943,sent=3951)
 /172.17.0.2:32876[1] (queued=0,recved=1702,sent=1702)
 /172.17.0.2:60538[1] (queued=0,recved=4624,sent=4631)
 /172.17.0.2:32858[1] (queued=0,recved=1953,sent=1953)
 /172.17.0.2:35346[1] (queued=0,recved=15272,sent=15272)
 /172.17.0.2:32812[1] (queued=0,recved=5548,sent=5582)
 /172.17.0.2:57688[1] (queued=0,recved=1659,sent=1659)
 /172.17.0.2:57974[1] (queued=0,recved=1656,sent=1656)
 /172.17.0.2:32822[1] (queued=0,recved=1596,sent=1596)
 /172.17.0.2:32816[1] (queued=0,recved=1599,sent=1599)
 /172.17.0.2:32884[1] (queued=0,recved=1597,sent=1597)
 /127.0.0.1:54606[0] (queued=0,recved=1,sent=0)
 /172.17.0.2:60536[1] (queued=0,recved=3211,sent=3211)

Latency min/avg/max: 0/5/16917
Received: 50350
Sent: 50406
Connections: 13
Outstanding: 0
Zxid: 0x7bb
Mode: standalone
Node count: 204
Connection closed by foreign host.
[root@sandbox ~]#
```

Figura 1.3.: Comprobación accesibilidad Zookeeper

```
[root@sandbox ~]# java -version
openjdk version "1.8.0_141"
OpenJDK Runtime Environment (build 1.8.0_141-b16)
OpenJDK 64-Bit Server VM (build 25.141-b16, mixed mode)
[root@sandbox ~]# echo $JAVA_HOME
/usr/lib/jvm/java
[root@sandbox ~]#
```

Figura 1.4.: Comprobación versión Java

de OpenTSDB.

Otro de los requisitos que deberían verificarse es que java se encuentra instalado, para ello y para conocer la versión que se tiene instalada se tecleará el comando «java -versión» y además para conocer el valor de la variable de entorno «JAVA_HOME», se ejecutará el comando «echo \$JAVA_HOME». Todo esto se puede ver en la figura 1.4. Se recomienda tener instalada el menos la versión 8 de Java, para poder ejecutar con éxito OpenTSDB.

Otra de las herramientas necesarias para poder instalar con éxito OpenTSDB es «git». Para ello se ejecutará el siguiente comando dentro de la pantalla del shell del sistema operativo CentOS:

```
yum install git
```

Una vez instalada la anterior herramienta, también es preciso instalar «development Tools», para ello se deberá ejecutar el siguiente comando:

```
yum groupinstall 'Development Tools'
```


1. Primeros pasos.

```
[root@sandbox ~]# git clone git://github.com/OpenTSDB/opentsdb.git
Initialized empty Git repository in /root/opentsdb/.git/
remote: Counting objects: 16000, done.
remote: Total 16000 (delta 0), reused 0 (delta 0), pack-reused 16000
Receiving objects: 100% (16000/16000), 31.36 MiB | 1.02 MiB/s, done.
Resolving deltas: 100% (9926/9926), done.
[root@sandbox ~]# ll
total 38404
-rw----- 2 root root    2439 Jun  2  2016 anaconda-ks.cfg
-rw-r--r-- 1 root root 474367 Jul 28 14:07 blueprint.json
-rw-r--r-- 1 root root     20 Jul 28 14:21 build.out
drwxr-xr-x 2 root root   4096 Jul 28 13:39 hdp
-rw-r--r-- 2 root root   7243 Jun  2  2016 install.log
-rw-r--r-- 2 root root   1680 Jun  2  2016 install.log.syslog
drwxr-xr-x 8 root root   4096 Nov 12 03:43 opentsdb
-rw-r--r-- 1 root root 38815340 Mar 18 2016 istudio-server-rhel-0.99.893-x86_64.rpm
-rw-r--r-- 1 root root    281 Jul 28 14:26 sandbox.info
lrwxrwxrwx 1 root root    48 Jul 28 14:08 start_ambari.sh -> /usr/lib/hue/tools/start_scripts/start_ambari.sh
lrwxrwxrwx 1 root root    47 Jul 28 14:08 start_hbase.sh -> /usr/lib/hue/tools/start_scripts/start_hbase.sh
```

Figura 1.5.: Carpeta generada al cargar OpenTDB

Una vez instaladas todas las aplicaciones anteriores, se procede a ejecutar el comando que permite bajar a nuestro equipo todos los paquetes necesarios para instalar OpenTSDB. Todos estos paquetes se bajan de «github» y se hace con el siguiente comando del sistema operativo:

```
git clone git://github.com/OpenTSDB/opentsdb.git
```

Para el desarrollo de este trabajo, se ha utilizado «Putty» para acceder al entorno de trabajo generado por HortonWorks. Cuando se accede con este sistema, el directorio en el que por defecto se encuentra es «/root». Al hacer esta descarga, se crea otra carpeta denominada opentsdb, tal y como se puede apreciar en la figura 1.5.

Ahora procede cambiar al directorio donde se han descargado todos los paquetes indicados anteriormente, se hace por lo tanto con el siguiente comando:

```
cd opentsdb
```

Al entrar en esta carpeta, podremos ver que se tiene una estructura de ficheros como la indicada en la figura 1.6.

Como puede verse, dentro de esa estructura de ficheros, así creada, se encuentra el fichero denominado «build.sh», que deberá ejecutarse para proceder a la instalación de OpenTSDB. Por lo tanto se deberá ejecutar el siguiente comando²:

```
./build.sh
```

Después de ejecutar este comando, en la pantalla del shell en el que se esté trabajando aparecerán una serie de instrucciones que de forma automática son ejecutadas por los comandos que se van ejecutando. Al final de todas las líneas de resultados generados, se podrá ver una pantalla similar a la que aparece en la figura 1.7 indicando de esta manera la correcta ejecución de todos los comandos.

²Asegurarse que se está dentro de la carpeta «opentsdb»

1. Primeros pasos.

```
[root@sandbox ~]# cd opentsdb
[root@sandbox opentsdb]# ll
total 196
-rw-r--r-- 1 root root 813 Nov 12 03:43 AUTHORS
-rwxr-xr-x 1 root root 740 Nov 12 03:43 bootstrap
drwxr-xr-x 4 root root 4096 Nov 12 03:43 build-aux
-rw-r--r-- 1 root root 221 Nov 12 03:43 build-bigtable.sh
-rw-r--r-- 1 root root 222 Nov 12 03:43 build-cassandra.sh
-rwxr-xr-x 1 root root 206 Nov 12 03:43 build.sh
-rw-r--r-- 1 root root 2278 Nov 12 03:43 configure.ac
-rw-r--r-- 1 root root 35147 Nov 12 03:43 COPYING
-rw-r--r-- 1 root root 26530 Nov 12 03:43 COPYING.LESSER
-rw-r--r-- 1 root root 32602 Nov 12 03:43 Makefile.am
-rw-r--r-- 1 root root 20140 Nov 12 03:43 NEWS
-rw-r--r-- 1 root root 2903 Nov 12 03:43 opentsdb.spec.in
-rw-r--r-- 1 root root 18539 Nov 12 03:43 pom.xml.in
-rw-r--r-- 1 root root 1024 Nov 12 03:43 README
drwxr-xr-x 15 root root 4096 Nov 12 03:43 src
drwxr-xr-x 16 root root 4096 Nov 12 03:43 test
-rw-r--r-- 1 root root 2918 Nov 12 03:43 THANKS
drwxr-xr-x 25 root root 4096 Nov 12 03:43 third_party
drwxr-xr-x 3 root root 4096 Nov 12 03:43 tools
-rw-r--r-- 1 root root 3235 Nov 12 03:43 tsdb.in
```

Figura 1.6.: Estructura de ficheros de OpenTSDB

```
/usr/lib/jvm/java/bin/java -Djava.util.prefs.userRoot=/root -cp `jar=third_party/
validation-api/validation-api-1.0.0.GA.jar; test -f "$jar" && echo "$jar" || echo "../$jar"
``:`jar=third_party/validation-api/validation-api-1.0.0.GA-sources.jar; test -f "$jar" && e
echo "$jar" || echo "../$jar"``:`jar=third_party/gwt/gwt-dev-2.6.0.jar; test -f "$jar" && e
cho "$jar" || echo "../$jar"``:`jar=third_party/gwt/gwt-user-2.6.0.jar; test -f "$jar" && e
cho "$jar" || echo "../$jar"``:`jar=third_party/gwt/opentsdb-gwt-theme-1.0.0.jar; test -f "
$jar" && echo "$jar" || echo "../$jar"``:../src com.google.gwt.dev.Compiler \
    -ea -strict -war gwt.tsd.QueryUi
Nov 12, 2017 4:06:58 AM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
Compiling module tsd.QueryUi
    Compiling 5 permutations
        Compiling permutation 0...
        Compiling permutation 1...
        Compiling permutation 2...
        Compiling permutation 3...
        Compiling permutation 4...
    Compile of permutations succeeded
Linking into /root/opentsdb/build/gwt/queryui
    Link succeeded
    Compilation succeeded -- 43.567s
/bin/mkdir -p staticroot
cp ../src/tsd/static/favicon.ico ../src/tsd/static/opentsdb_header.jpg staticroot
find -L staticroot -type l -exec rm {} \;
p=`pwd`/gwt/queryui && cd staticroot \
    && for i in $p/*; do ln -s -f "$i" || break; done
find -L staticroot/gwt -type f | xargs touch
make[1]: Leaving directory `/root/opentsdb/build'
```

Figura 1.7.: Comandos obtenidos en la instalación.

1. Primeros pasos.

```
/usr/bin/install -c -m 644 'gwt/opentsdb/images/ie6/hborder_blue_shadow.png' '/usr/local/share/opentsdb/static/gwt/opentsdb/images/ie6/hborder_blue_shadow.png'
/usr/bin/install -c -m 644 'gwt/opentsdb/images/splitPanelThumb.png' '/usr/local/share/opentsdb/static/gwt/opentsdb/images/splitPanelThumb.png'
/usr/bin/install -c -m 644 'gwt/opentsdb/images/corner.png' '/usr/local/share/opentsdb/static/gwt/opentsdb/images/corner.png'
/usr/bin/install -c -m 644 'gwt/opentsdb/opentsdb_rtl.css' '/usr/local/share/opentsdb/static/gwt/opentsdb/opentsdb_rtl.css'
/usr/bin/install -c -m 644 'gwt/opentsdb/opentsdb.css' '/usr/local/share/opentsdb/static/gwt/opentsdb/opentsdb.css'
/usr/bin/install -c -m 644 'favicon.ico' '/usr/local/share/opentsdb/static/favicon.ico'
/usr/bin/install -c -m 644 '9C0584C81E4C0C44CE7F22E963B8F9F8.cache.html' '/usr/local/share/opentsdb/static/9C0584C81E4C0C44CE7F22E963B8F9F8.cache.html'
/usr/bin/install -c -m 644 'C049F9D968CB6A1F5EB0663039470155.cache.html' '/usr/local/share/opentsdb/static/C049F9D968CB6A1F5EB0663039470155.cache.html'
/usr/bin/install -c -m 644 'hosted.html' '/usr/local/share/opentsdb/static/hosted.html'
/usr/bin/install -c -m 644 'B8E8AB8803042A1F67F38F1B2548575E.cache.html' '/usr/local/share/opentsdb/static/B8E8AB8803042A1F67F38F1B2548575E.cache.html'
/usr/bin/install -c -m 644 'clear.cache.gif' '/usr/local/share/opentsdb/static/clear.cache.gif'
/usr/bin/install -c -m 644 '3E2454C2D466DFD13893E2ED448CA684.cache.html' '/usr/local/share/opentsdb/static/3E2454C2D466DFD13893E2ED448CA684.cache.html'
test -z "/usr/local/share/opentsdb/static" || /bin/mkdir -p "/usr/local/share/opentsdb/static"
/usr/bin/install -c -m 644 ../src/tsd/static/favicon.ico ../src/tsd/static/opentsdb_header.jpg '/usr/local/share/opentsdb/static'
make[2]: Leaving directory `/root/opentsdb/build'
make[1]: Leaving directory `/root/opentsdb/build'
```

Figura 1.8.: Salida del comando «make install»

```
[root@sandbox build]# whereis tsdb
tsdb: /usr/local/bin/tsdb
```

Figura 1.9.: Localización de TSDB

Ahora faltaría completar la instalación de OpenTSDB y para ello hay que ir a la carpeta contenida dentro de «opentsdb» y denominada «built», para ello hay que ejecutar el siguiente comando.

```
cd /root/opentsdb/build
```

y una vez colocado en esta carpeta ejecutar el siguiente comando:

```
make install
```

Igualmente nos genera una serie de salidas de ejecución del código, y como final de esas salidas, se obtiene lo que se observa en la figura 1.8.

De esta manera, ya se tendría instalado OpenTDB listo para poder trabajar con él, aunque eso sí aún quedan algunos ajustes con el cluster de HBase que se expondrán y detallarán a continuación. Para saber dónde se tiene instalado OpenTSDB, podemos ejecutar los comandos que se muestran en la figura 1.9.

1.2.1.1. Configuración de OpenTSDB con HBase.

Con las instrucciones dadas hasta este momento, se tiene instalado y ejecutándose a OpenTSDB, sin embargo se necesita hacer una configuración mínima para que interaccio-

1. Primeros pasos.

ne adecuadamente con el cluster Hbase que se tiene instalado, en este caso sobre HortonWorks. Se insiste en que es una configuración mínima, ya que OpenTSDB posee muchos parámetros de configuración que serán expuestos más adelante en el apartado [1.4 en la página 17](#). El fichero de configuración se denomina «opentsdb.conf» y para la instalación que aquí se ha desarrollado, se localizará dentro de la carpeta «/root/opentsdb/src/». Los parámetros que en un principio se modificarán serán los siguientes³.

- **tsd.network.port**. Es el puerto por donde atenderá peticiones OpenTSDB. En este caso se le dará un valor de «8091», ya que es un puerto libre que por defecto ya tiene preinstalado HortonWorks⁴.
- **tsd.http.cachedir**. Es lugar donde se almacenarán los fichero temporales de la caché. En este tutorial, se indicará el valor de «/tmp».
- **tsd.http.staticroot**. Es el lugar donde se almacenarán los ficheros estáticos, por ejemplo los ficheros de JavaScript que generan la GUI de OpenTSDB. En este caso el valor que se asignará será «/root/opentsdb/build/staticroot».
- **tsd.storage.hbase.zk_quorum**. Es una lista de hosts separados por comas, donde se puede localiza a ZooKeeper. En nuestro caso y dado que se tiene una instalación local de HortonWorks, al valor que se asignará a este parámetro será «localhost:2181»⁵.

Una vez hecha esta configuración, se necesitan crear una serie de tablas que utiliza OpenTSDB dentro de HBase, para ello se utilizara el siguiente comando genérico:

```
env COMPRESSION=NONE HBASE_HOME=path/to/hbase-0.94.X ./src/create_table.sh
```

Donde habrá que concretar el camino donde se tenga instalado HBase. En este caso concreto el valor será «/usr/hdp/2.6.1.0-129/hbase»⁶. Por lo tanto el comando concreto que se deberá ejecutar en la pantalla de comandos será el siguiente⁷:

```
env COMPRESSION=NONE HBASE_HOME=/usr/hdp/2.6.1.0-129/hbase ./src/create_table.sh
```

Después de ejecutar estos comandos, la ventana de comandos del sistema operativo nos indica la creación de cuatro tablas en HBase, tal y como se puede ver en la figura [1.10](#).

Si ahora se entra en el shell de HBase (con la instrucción de la línea de comandos hbase shell) podremos ver con la instrucción de HBase «list» que efectivamente se han creado las tablas, tal y como se puede ver en la figura [1.11](#)

³La modificación se hará utilizando el editor que se desee, por ejemplo Vi, Vim o nano

⁴Las comillas no se pondrán en el fichero opentsdb.conf. Este inciso también es válido para el resto de parámetros que a continuación de detallan.

Si se quiere utilizar otro número de puerto diferente al que por defecto ya viene instalado con HortonWorks, se deberá hacer de acuerdo a las instrucciones marcadas en la documentación de HortonWorks, y que [se pueden ver en este enlace](#).

⁵El valor 2181 es el puerto que tiene ZooKeeper por defecto

⁶Lógicamente este valor puede cambiar dependiendo de dónde se tenga la instalación de Hbase

⁷Se deberá estar en el raíz de la instalación de Hbase, es decir en este caso en «/root/opentsdb/»

1. Primeros pasos.

```
[root@sandbox opentsdb]# env COMPRESSION=NONE HBASE_HOME=/usr/hdp/2.6.1.0-129/hbase ./src/create_table.sh
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.6.1.0-129, r718c773662346de98a8ce6fd3b5f64e279cb87d4, Wed May 31 03:27:31 UTC 2017

create 'tsdb-uid',
  (NAME => 'id', COMPRESSION => 'NONE', BLOOMFILTER => 'ROW'),
  (NAME => 'name', COMPRESSION => 'NONE', BLOOMFILTER => 'ROW')
0 row(s) in 1.6410 seconds

Hbase::Table - tsdb-uid

create 'tsdb',
  (NAME => 't', VERSIONS => 1, COMPRESSION => 'NONE', BLOOMFILTER => 'ROW')
0 row(s) in 1.2530 seconds

Hbase::Table - tsdb

create 'tsdb-tree',
  (NAME => 't', VERSIONS => 1, COMPRESSION => 'NONE', BLOOMFILTER => 'ROW')
0 row(s) in 2.2390 seconds

Hbase::Table - tsdb-tree

create 'tsdb-meta',
  (NAME => 'name', COMPRESSION => 'NONE', BLOOMFILTER => 'ROW')
0 row(s) in 2.2400 seconds

Hbase::Table - tsdb-meta
[root@sandbox opentsdb]#
```

Figura 1.10.: Tablas generadas en HBase

```
Hbase::Table - tsdb-meta
[root@sandbox opentsdb]# hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.1.2.6.1.0-129, r718c773662346de98a8ce6fd3b5f64e279cb87d4, Wed May 31 03:27:31 UTC 2017

hbase(main):001:0> list
TABLE
ATLAS_ENTITY_AUDIT_EVENTS
atlas_titan
iemployee
t1
tsdb
tsdb-meta
tsdb-tree
tsdb-uid
8 row(s) in 0.2350 seconds

=> ["ATLAS_ENTITY_AUDIT_EVENTS", "atlas_titan", "iemployee", "t1", "tsdb", "tsdb-meta", "tsdb-tree", "tsdb-uid"]
```

Figura 1.11.: Tablas vistas desde HBase.

1. Primeros pasos.

```
# Path under which the znode for the -ROOT- region is located, default is "/hbase"  
#tsd.storage.hbase.zk_basedir = /hbase
```

Figura 1.12.: Localización de znode for the -Root-region

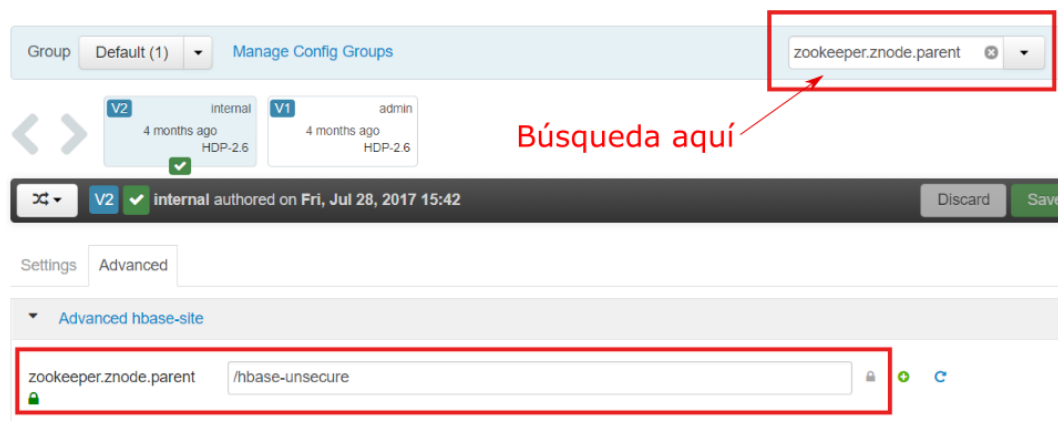


Figura 1.13.: Valores de zookeeper.znode.parent en HBase

En estos momentos, ya se tendría instalado y configurado OpenTSDB de tal manera que el servidor se podría arrancar con el siguiente comando:

```
./build/tsdb tsd --config=/root/opentsdb/src/opentsdb.conf
```

Observar que en el anterior comando se ha indicado la ubicación del fichero de configuración para que tome los valores de los parámetros en el arranque del mismo. Si hacemos el arranque de esta manera, podremos observar que nos genera el siguiente error: «2017-11-12 07:04:27,584 ERROR [main-EventThread] HBaseClient: The znode for the -ROOT- region doesn't exist! ». Ello es debido a que no hay sincronización entre los parámetros que hemos definido aquí y los que posee HBase en la instalación que por defecto se hace en HortonWorks (este error puede verse explicado en <https://github.com/OpenTSDB/opentsdb/issues/723>).

En efecto, si miramos en el fichero de configuración de OpenTSDB, se puede ver lo que se muestra en la figura 1.12, donde puede apreciarse que el valor por defecto que se asigna en OpenTSDB para «tsd.storage.hbase.zk_basedir» es «/hbase».

Sin embargo, si nos vamos a la configuración de HBase que nos ofrece HortonWorks y que se encuentra localizado en la variable de configuración de HBase denominada «zookeeper.znode.parent», podremos observar que el valor es diferente, tal y como se puede observar en la figura 1.13.

Estos dos parámetros deben coincidir y para ello lo que se ha hecho es modificar el valor del parámetro de configuración de OpenTSDB de «tsd.storage.hbase.zk_baseir» a «/hbase-unsecure» con la finalidad de que tengan el mismo valor en los dos lugares.

Realizados estos cambios, se ejecuta de nuevo OpenTSD con la instrucción dada anteriormente, y podemos comprobar que ahora sí el servidor de OpenTSDB se pone a la

1. Primeros pasos.

```
st: sandbox.hortonworks.com is: 172.17.0.2
2017-11-12 08:11:43,356 INFO [AsyncHBase I/O Worker #1] HBaseClient: Added client for region Re
gionInfo(table="tsdb-uid", region_name="tsdb-uid,,1510468731097.859d9ffef12a75493dc70590d7e6c6a3
.", stop_key=""), which was added to the regions cache. Now we know that RegionClient@384416485
(chan=[id: 0x8c21b06, /172.17.0.2:45224 => /172.17.0.2:16020], #pending_rpc=0, #batched=0, #rp
cs_inflight=1) is hosting 2 regions.
2017-11-12 08:11:43,380 INFO [main] RpcManager: Mode: rw, HTTP UI Enabled: true, HTTP API Enabl
ed: true
2017-11-12 08:11:43,407 INFO [main] RpcHandler: TSD is in rw mode
2017-11-12 08:11:43,407 INFO [main] RpcHandler: CORS domain list was empty, CORS will not be en
abled
2017-11-12 08:11:43,408 INFO [main] RpcHandler: Loaded CORS headers (Authorization, Content-Typ
e, Accept, Origin, User-Agent, DNT, Cache-Control, X-Mx-ReqToken, Keep-Alive, X-Requested-With,
If-Modified-Since)
2017-11-12 08:11:43,410 INFO [main] ConnectionManager: TSD concurrent connection limit set to:
0
2017-11-12 08:11:43,413 WARN [main] PluginLoader: Unable to locate any plugins of the type: net
.opentsdb.tsd.HttpSerializer
2017-11-12 08:11:43,431 INFO [main] TSDMain: Ready to serve on /0.0.0.0:8091
```

Figura 1.14.: Servidor OpenTSDB escuchando.

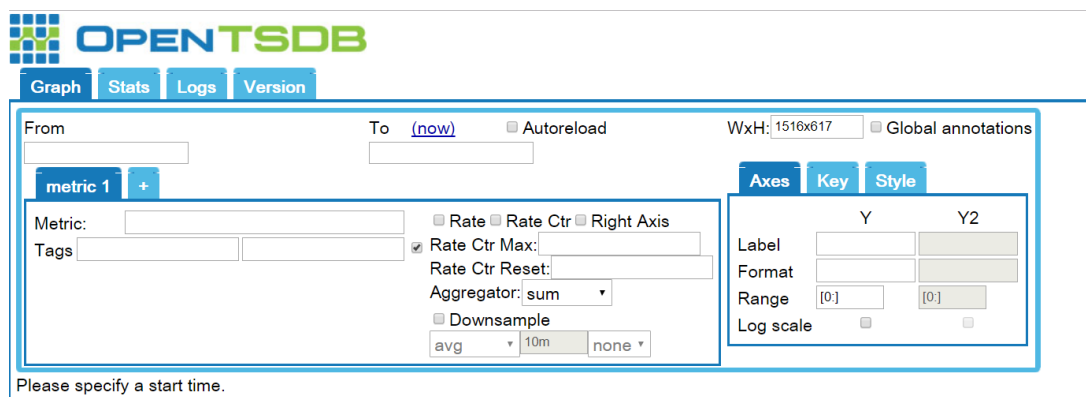


Figura 1.15.: GUI de OpenTSDB

escucha, para recibir ordenes desde cualquier cliente. La situación en la que se queda la ventana de comandos se muestra en la figura 1.14.

Y ahora ya se tendría todo listo, para poder ver el GUI que por defecto tiene OpenTSDB. Para ello desde cualquier explorador, se pone en la barra de direcciones lo siguiente: «localhost:8091» y se aparecerá dicho GUI tal y como se muestra en la figura 1.15.

Para concluir este apartado, simplemente añadir que para poner en marcha el servidor sin necesidad de incluir siempre el fichero que contiene la configuración de OpenTSDB, lo que el autor de este trabajo ha hecho, es crear un fichero denominado «opentsdb_exe.sh» que contiene la instrucción comentada anteriormente para levantar el servidor OpenTSDB. Después de han cambiado los permisos de este fichero para que al menos se pueda ejecutar, y listo tan sólo hay que escribir el comando «opentsdb_exe.sh» para levantar el servidor de una forma fácil y sencilla.

1.2.2. Instalación utilizando Ambari.

Este apartado quedaría por hacer ya que la instalación que se hace con Ambari tiene el inconveniente que después no se puede eliminar. Los enlaces que sirven para esto son los siguientes:

<https://community.hortonworks.com/articles/4577/use-opentsdb-to-storevisualize-stock-data-on-hdp-s.html>

Otro enlace: <https://github.com/hortonworks-gallery/ambari-opentsdb-service>

1.3. Primer ejemplo con OpenTSDB.

En los apartados anteriores se han expuesto los pasos necesarios para poder poner en funcionamiento el servidor OpenTSDB. Por lo tanto ya se está en disposición de poder desarrollar algún ejemplo que permita almacenar datos en las tablas que se han creado durante la instalación de OpenTSDB.

Se van a utilizar para este primer ejemplo datos de cotización en bolsa de una serie de valores. Para bajar los datos, se utilizará el API que proporciona Google Finance y que de una forma escueta se comenta en el apéndice **B en la página 70**. Gracias a esta metodología, se ha podido escribir el script que se muestra en el apéndice **A en la página 66** mediante el cual se va a poder bajar los datos que se necesitan para el ejemplo que se muestra en este apartado.

En el directorio «/root/opentsdb/» se ha creado la carpeta «example» en la cual se van a colocar todos los datos necesarios para realizar este primer ejemplo. En primer lugar lo que se hará utilizando cualquier editor de textos (VI, Vim, nano, etc.) es crear en el directorio «/root/opentsdb/example» el fichero «google_intraday.py» y en el pegar el código que figura en el anexo **A en la página 66**. Una vez se tenga este fichero en funcionamiento, se deberán ejecutar los siguientes comandos teniendo en cuenta que se debe estar en la carpeta «/root/opentsdb/example»:

```
rm -f prices.csv
rm -f opentsd.input

python google_intraday.py AAPL > prices.csv
python google_intraday.py GOOG >> prices.csv
python google_intraday.py HDP >> prices.csv
python google_intraday.py ORCL >> prices.csv
python google_intraday.py MSFT >> prices.csv
```

Al final de este proceso, se deberá tener en la carpeta «/root/opentsdb/example/» la estructura que se muestra en la figura **1.16**.

Si después de este proceso, se ejecuta en el sistema operativo el comando «tail opentsd.input», se puede ver la estructura del fichero final gracias a los datos que aparecen al final del mismo. Esta estructura se muestra en la figura **1.17**

Todo queda listo para poder exportar los datos a la base de datos OpenTSDB, para conseguirlo, se deberá ejecutar el comando de OpenTSDB que sigue y que en capítulos posteriores se explicará con mayor detalle su funcionamiento:

1. Primeros pasos.

```
[root@sandbox ~]# cd opentsdb/example/
[root@sandbox example]# ll
total 5484
-rwxrwxrwx 1 root root    5812 Nov 12 15:44 google_intraday.py
-rw-r--r-- 1 root root 4226601 Nov 12 15:49 opentsd.input
-rw-r--r-- 1 root root 1377414 Nov 12 15:49 prices.csv
[root@sandbox example]#
```

Figura 1.16.: Estructura carpeta example

```
[root@sandbox ~]# cd opentsdb/example/
[root@sandbox example]# tail opentsd.input
open 1512766740 84.105 symbol=MSFT
close 1512766740 84.11 symbol=MSFT
high 1512766740 84.13 symbol=MSFT
low 1512766740 84.08 symbol=MSFT
volume 1512766740 183323 symbol=MSFT
open 1512766800 84.11 symbol=MSFT
close 1512766800 84.16 symbol=MSFT
high 1512766800 84.18 symbol=MSFT
low 1512766800 84.1 symbol=MSFT
volume 1512766800 2394965 symbol=MSFT
[root@sandbox example]#
```

Figura 1.17.: Estructura datos del ejemplo

```
/root/opentsdb/build/tsdb import opentsd.input --zkbasedir=/hbase-
unsecure --zkquorum=localhost:2181 --auto-metric --config=/root/
opentsdb/src/opentsdb.config
```

Sin embargo cuando se va a utilizar por primera vez la GUI de OpenTSDB y después de introducir los valores correspondientes en los campos de la misma, aparece el error «nice: gnuplot: No such file or directory». Ello es debido a que por defecto no está instalado el programa «gnuplot». Por lo tanto se deberá proceder a instalar este programa, utilizando para ello el siguiente comando en la ventana de comandos del sistema operativo «yum install gnuplot». Una vez instalado este programa desaparecerá el error, y podremos ver la salida en la GUI de openTSDB, tal y como se muestra en la figura 1.18.

En este apartado sacamos a título orientativo una pequeña muestra de lo que se puede obtener utilizando la GUI de OpenTSDB. En un capítulo posterior, se utilizará todo un apartado destinado a esta herramienta. No obstante también conviene decir que existe otra herramienta gráfica mucho más potente que la que ofrece OpenTSDB y que se denomina Grafana. El lector interesado, puede encontrar más información sobre Grafana haciendo clic en el siguiente enlace: <https://grafana.com/>.

1.4. Configuración de OpenTSDB.

La base de datos OpenTSDB cuenta con un importante conjunto de comandos que permiten configurar a la base de datos para adaptarse a muchas circunstancias y entornos diferentes. Estos parámetros de configuración pueden estar bien en un fichero de confi-

1. Primeros pasos.



Figura 1.18.: Salida de OpenTSDB para el ejemplo.

1. Primeros pasos.

guración (el denominado «opentsdb.conf» que ya se ha utilizado anteriormente) o bien mediante el uso de comando o bien una mezcla de los dos casos citados anteriormente.

El fichero de configuración está constituido por una serie de líneas del tipo clave-valor todos ellos escritos en letras minúsculas. La clave es el nombre de la propiedad que se quiere configurar, le sigue un signo igual y a continuación está el valor concreto que se quiere asignar a esa propiedad. Todos los nombres de las propiedades usadas en OpenTSDB comienzan con la expresión «tsd.», y los comentarios que figuran en ese fichero están indicados con el símbolo «#». Un ejemplo concreto de esta representación ya se ha mostrado anteriormente en la figura 1.12 en la página 14.

Dado que las propiedades se pueden indicar tanto en un fichero, como en la línea de comandos o con una mezcla de ambos, el orden de prioridad que sigue OpenTSDB es el siguiente:

1. Inicialmente se cargan los valores que existen por defecto.
2. Después se cargan los valores de configuración existentes en el fichero «opentsdb.conf». Sobrescribiendo los valores existentes por defecto.
3. Por último se se cargan los valores asignados en la línea de comandos, sobrescribiendo los valores por defecto y los del fichero de configuración.

La ubicación del fichero de configuración se puede indicar con el argumento «--config» que acompaña a muchos comandos escritos en la línea de comandos, debiendo acompañar a este argumento y después del signo igual el camino completo que indica la ubicación del fichero «opentsdb.conf». Si no se especifica este argumento, entonces OpenTSDB buscará el fichero de configuración en las siguientes ubicaciones:

- ./opentsdb.config
- /etc/opentsdb.conf
- /etc/opentsdb/opentsdb.conf
- /opt/opentsdb/opentsdb.conf

A continuación se muestran algunos de los valores que se pueden asignar en el fichero de configuración. Una lista completa de los mismos se puede encontrar en la documentación de OpenTSDB que se puede visualizar haciendo clic en este enlace: <http://opentsdb.net/docs/build/html/using>

- **tsd.core.auto_create_metrics**. Para indicar si a un «data point» con un nueva métrica se le asigna un «UID». Si el valor es «false» un nuevo «data point» con una nueva métrica es rechazado y se genera una excepción. **Type:** Boolean; **Requerido:** Opcional; **Default:** False; **CLI:** --auto-metric.
- **tsd.core.auto_create_tags (2.1)**. Para indicar si un nuevo «data point» con un nombre de tag (o en castellano: etiqueta) también nuevo o no existente anteriormente, se le asigna un nuevo UID a ese nombre de tag. Si se le asigna un valor de «false» entonces el «data point» va a ser rechazado y se produciría una excepción. **Type:** Boolean; **Requerido:** Opcional; **Default:** True.

1. Primeros pasos.

- **tsd.core.auto_create_tagvs** (2.1). Para indicar si un nuevo «data point» con un valor nuevo de un tag se le asigna o no un nuevo UID. Si el valor de esta propiedad es «false» entonces, cuando se produce esta circunstancia se rechaza el «data point» y se genera un excepción. **Type:** Boolean; **Requerido:** Opcional; **Default:** True.
- **tsd.core.connections.limit** (2.3). Para indicar el número máximo de conexiones que se permitirá a TSD. Si se supera este límite, entonces las nuevas conexiones serán rechazadas. **Type:** Integer; **Requerido:** Opcional; **Default:** 0.
- **tsd.core.enable_api** (2.3). Con esta opción se permite o no que el HTTP API 2.x funcione. Cuando está deshabilitado, ciertas llamadas del HTTP API como /api/query o /api/suggest van a devolver un error de tipo 404. **Type:** Boolean; **Requerido:** Opcional; **CLI:** --disable-api.
- **tsd.core.enable_ui** (2.3). Se elige esta opción para indicar si se habilitamos o no la visibilidad del GUI de Opentsdb. Cuando está deshabilitado, instrucciones del tipo /logs o /suggest devolverán un error de tipo 404. **Type:** Boolean; **Requerido:** Opcional; **CLI:** --disable-ui .
- **tsd.core.meta.enable_realtime_ts**. Si se activa o no la creación de objetos «real-time TSMeta». **Type:** Boolean; **Requerido:** Opcional; **valor-defecto:** False .
- **tsd.core.meta.enable_realtime_uid**. Si se activa o no la creación de objetos «real-time IDMeta». **Type:** Boolean; **Requerido:** Opcional; **valor-defecto:** False .
- **tsd.core.meta.enable_tsuid_incrementing**. Si se habilita o no un seguimiento de los TSUIDs mediante el incremento de un contador cada vez que un «data point» es recogido. **Type:** Boolean; **Requerido:** Opcional; **valor-defecto:** False .
- **tsd.core.meta.enable_tsuid_tracking**. Si se habilita o no un seguimiento de los TSUIDs almacenando un 1 con el actual «timestamp» cada vez que se recoge un nuevo «data point». **Type:** Boolean; **Requerido:** Opcional; **valor-defecto:** False .
- **tsd.core.plugin_path**. Sería el path que se utiliza cuando comienza TSD. Si hay algún fallo en el path, TSD no comienza. **Type:** Boolean; **Requerido:** Opcional.
- **tsd.core.preload_uid_cache** (2.1). Establece o no la pre-carga de las caches de los UID cuando comienza TSD. **Type:** Boolean; **Requerido:** Opcional; **valor-defecto:** False .
- **tsd.core.preload_uid_cache_max_entries** (2.1). Número de filas a escanear en la precarga de UID. **Type:** Integer; **Requerido:** Opcional; **valor-defecto:** 300.000 .
- **tsd.core.timezone**. Un identificador de un «timezone» para trabajar de acuerdo a esa información con las fechas de los «data points». **Type:** String; **Requerido:** Opcional; **valor-defecto:** Según config. del sistema .

1. Primeros pasos.

- **tsd.core.tree.enable_processing**. Si se habilita o no procesamiento de TSMeta, mediante las reglas de estructura de árbol. **Type**: Boolean; **Requerido**: Opcional; **valor-defecto**: false .
- **tsd.core.uid.random_metrics** (2.2). Si se asigna o no una asignación aleatoria de UUIDs a las nuevas métricas cuando son creadas. **Type**: Boolean; **Requerido**: Opcional; **valor-defecto**: false .
- **tsd.http.cachedir**. Para indicar el camino completo donde se van a colocar los ficheros temporales. **Type**: Boolean; **Requerido**: Opcional; **CLI**: --cachedir.
- **tsd.http.query.allow_delete**. Para indicar si se permite o no borrar «data points» desde una query. **Type**: Boolean; **Requerido**: Opcional; **valor-defecto**: false .
- **tsd.http.show_stack_trace**. Para indicar si se devuelve o no la pila de errores (también denominado stack trace) cuando ocurre una exception. **Type**: Boolean; **Requerido**: Opcional; **valor-defecto**: false .
- **tsd.http.staticroot**. Localización del lugar donde se van a almacenar los ficheros estáticos (como por ejemplo javascript) que utiliza la aplicación web que tiene OpenTSDB. **Type**: String; **Requerido**: SI; **CLI**: --staticroot.
- **tsd.mode**. Si se permite o no escribir «data points». **Type**: String; **Requerido**: Opcional; **valor-defecto**: rw .
- **tsd.network.keep_alive**. Si se permiten o no conexiones del tipo «keep-alive». **Type**: Boolean; **Requerido**: Opcional; **valor-defecto**: True .
- **tsd.network.port**. El TCP utilizado para aceptar conexiones. **Type**: Integer; **Requerido**: SI; **CLI**: --port.
- **tsd.query.allow_simultaneous_duplicates** (2.2). Si se admiten o no queries simultáneas e iguales provenientes del mismo host. Si no se admiten, la segunda query generará una exception. **Type**: Boolean; **Requerido**: Opcional; **valor-defecto**: false .

2. Estructura de OpenTSDB.

Una vez hecha la presentación y puesta en marcha de OpenTSDB en el apartado anterior, en el presente se van a exponer y desarrollar conceptos de indudable interés para poder comprender todo el sistema de funcionamiento de la base de datos de OpenTSDB. En los siguientes apartados se va a proceder a mostrar una serie de conceptos y definiciones muy importantes para entender apartados posteriores.

2.1. Agregación.

Una de las grandes prestaciones que ofrece OpenTSDB es su facilidad para poder agregar datos de varios «data-points» en uno sólo. Los datos originales siguen estando almacenados en su formato original, pero sin embargo se pueden agregar para obtener series temporales de acuerdo a las necesidades de los usuarios finales. Para agrupar estos datos, existen las denominadas «funciones de agregación» tales como SUM, MEAN, etc.

Cuando se procede a hacer una agrupación los tiempos de cada una de las series se agrupan por tiempo, y a los valores correspondientes para ese tiempo se les aplica la correspondiente función de agregación. Si se hace un símil con el lenguaje SQL, la instrucción correspondiente sería similar a la siguiente: `SELECT SUM(value) FROM ts_table GROUP BY timestamp`.

2.2. Downsampling.

La ventaja que presenta OpenTSDB, es que puede almacenar una ingente cantidad de información. Además los datos de la serie temporal pueden estar tomados en periodos de tiempo hasta de milésimas de segundos si se quiere. Ello conlleva que cuando se realiza un query, pueden obtenerse como respuesta tanta cantidad de información, que la respuesta sea lenta, en incluso la representación de los datos obtenidos en la consulta puede ser hasta ilegible, sin que aporte ninguna información relevante el gráfico en cuestión. Un ejemplo de esta situación se puede ver en la figura 2.1.

Con la filosofía «Downsampling» de OpenTSDB se pueden reducir considerablemente el número de puntos obtenidos en la consulta y así evitar los inconvenientes comentados anteriormente para estos casos. Para poder ejecutar un Downsampling, se requiere definir una **función de agregación** y un **intervalo temporal**. Con la función de agregación, lo que se consigue es agrupar los datos y obtener unos solo, en base a los datos que figuran en el intervalo considerado. Los valores más comunes de la función de agregación son «sum» o «avg». En el supuesto de que la función elegida sea «sum» lo que se hace es

2. Estructura de OpenTSDB.

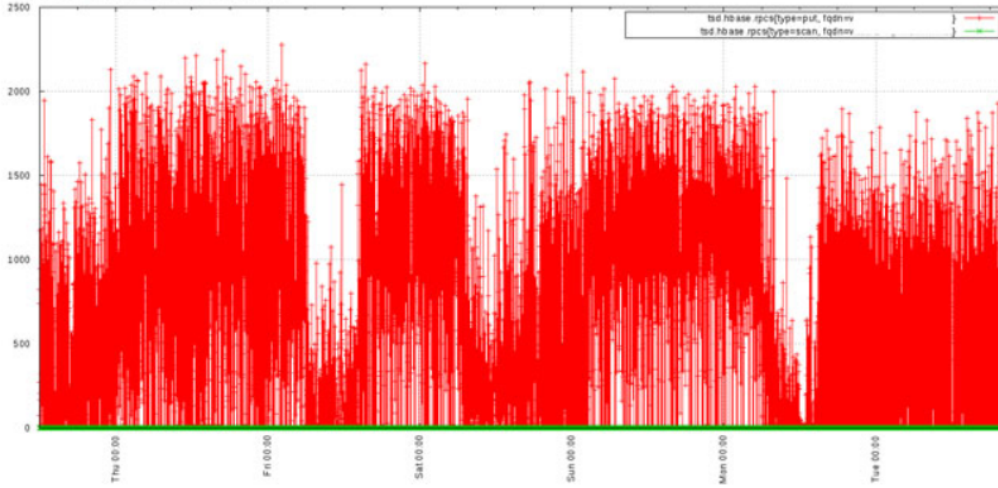


Figura 2.1.: Ejemplo de aglomeración de datos.

sumar los datos de los puntos de cada intervalo, para obtener un solo punto por intervalo temporal.

Los intervalos de tiempo se tienen el siguiente formato: «<cantidad><unidades>», de tal manera que con la expresión «1h» se hará referencia a 1 hora (se agruparán los datos por hora, utilizando para ello la función de agregación indicada). La expresión «30m» se referirá a 30 minutos y así para diferentes unidades de medida del tiempo. En este sentido, si se indica por ejemplo «30s-sum», se estará indicando que agrupe los datos en periodos de 30 segundos y proceda a su suma.

De esta manera, el gráfico mostrado en la figura 2.1, se puede hacer mucho más interpretable si se utiliza este procedimiento, de tal forma que se podría obtener como resultado final lo que se muestra en la figura 2.2.

2.3. Rate (o Razón).

En ocasiones, no interesa estudiar las series temporales en base a la información que facilitan los valores absolutos de la serie temporal, sino que más bien lo que interesa es el incremento o decremento relativo mediante tasas o razones. Para conseguir esto, OpenTSDB presenta la herramienta denominada «rate», que lo que calcula es una especie de derivada de la curva obtenida con los valores y en este sentido, el valor que calcula lo hace con la siguiente fórmula: $(v_2 - v_1)/(t_2 - t_1)$. De esta forma lo que se calcula es la razón del cambio por segundo.

2. Estructura de OpenTSDB.

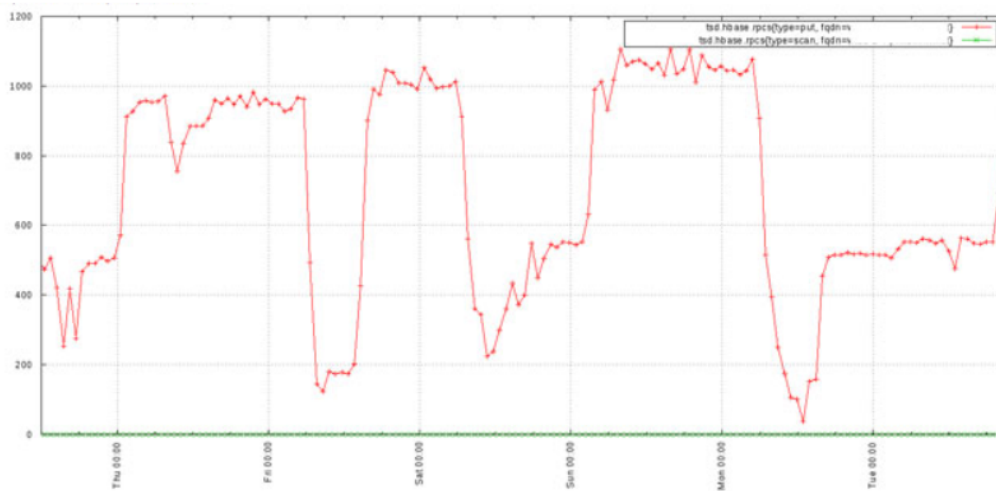


Figura 2.2.: Ejemplo de datos agrupados.

2.4. Filtros.

Todas las series temporales que se obtienen utilizando OpenTSDB están constituidas por una métrica y uno o más pares clave-valor (tags). En OpenTSDB, los filtros son aplicados sobre los valores de los tags. En este sentido, si en una query tan solo se hace constar sólo el nombre de la métrica, se obtendrán todos los «data point» asignados a esa métrica, sin tener en cuenta ningún par «clave-valor». Por hacer una analogía, los filtros en OpenTSDB son similares a los predicados «where» en una consulta SQL.

2.5. Rollup's y Pre-aggregates.

Rollup es otro término muy utilizado por OpenTSDB, que es esencia es lo mismo que el concepto de downsampling ya comentado en el apartado 2.2 en la página 22, por lo que no se insistirá más sobre este concepto.

El concepto de Rollup es muy útil cuando los datos almacenados son tan densos, que la información que proporcionan no está nada clara, debido precisamente a su intensidad. Obtener los valores mediante la función de agregación que se indique, puede ralentizar mucho la obtención de resultados. Por ese motivo, OpenTSDB ha incorporado en su estructura un nuevo concepto que es denominado «Pre-aggregates». En este caso lo único que se necesita es una función de agregación (no es necesario definir un periodo temporal como era el caso de los rollup's).

2.6. Componentes de una query.

2.7. UUIDs y TSUIDs.

Como ya se ha comentado en la figura 1.17 en la página 17, los denominados «data point»¹ tienen una estructura consistente en valor para la métrica, seguido de un valor para el tiempo, el valor que se mide en ese momento y después pares de tipo nombre-valor que son denominados tags.

De esta manera en OpenTSDB cuando se escribe un «data point» siempre se asocia con una métrica y al menos un tag clave-valor. Cada métrica, tag-nombre (denominado tagk) y tag-valor (denominado tagv) la primera vez que aparecen en la base de datos se les asigna un identificador único². La combinación de una métrica y uno o varios pares tag nombre-valor crean una serie temporal que es denominada TSUID.

En OpenTSDB los valores «pre-aggregates» se diferencian del resto, porque de forma automática se crea un tag (clave-valor), de tal manera que la clave se denomina «_aggregate»³ seguido del signo igual y en mayúsculas el nombre de la función de agregado utilizada. Un ejemplo puede ser el siguiente:

Serie ID	Métrica	Tag 1	Tag 2
ts1	system.ope.dat1	colo=lga	_aggregate=SUM
ts2	system.ope.dat2	colo=lga	_aggregate=COUNT
ts3	system.ope.dat3	colo=sjc	_aggregate=SUM
ts4	system.ope.dat3	colo=sjc	_aggregate=COUNT

2.7.1. UUID.

Como ya se ha indicado en apartados anteriores cada vez que se incorpora un «data point» a OpenTSDB, se incorpora el nombre de una métrica y una serie de tags (con formato «clave=valor»). Cuando se incorpora una nueva métrica o una nueva clave o un nuevo valor, se asigna a los mismos un único identificador (UUID).

Los diferentes tipos de objetos UUIDs que se pueden dar en OpenTSDB son los siguientes :

- **metric.**
- **tagk.** Con esta notación, OpenTSDB se refiere a la clave de los tags utilizados.
- **tagv.** Con esa notación, OpenTSDB se refiere al valor de los tags utilizados.

El valor UUID es un entero positivo que es único para cada uno de los nombres de los tres tipos que se han detallado anteriormente. En este sentido existe un contador para cada

¹Hay que tener cuidado para no confundir un «data point», con una serie temporal, ya que si en ocasiones los dos conceptos pueden coincidir, no siempre es así pues una serie temporal está constituida por el cruce de una métrica y uno o varias parejas de tags.

²Este identificador también se puede asignar vía API o mediante una herramienta CLI

³Esta clave es la que se usa por defecto, se puede cambiar en el fichero de configuración de OpenTSDB, utilizando para ello el valor definido por la variable «tsd.rollups.aggg_tag_key».

2. Estructura de OpenTSDB.

uno de esos tres tipos que se va incrementando cada vez que se incorpora un nuevo valor de esos tres tipos.

2.7.2. TSUID.

Cuando un «data point» es guardado sobre la base de datos OpenTSDB, la fila correspondiente es formateada de la siguiente manera:

```
<metrica_UID><timestamp><tagk1_UID><tagv1_UID> [...<tagkN_UID><tagvN_UID>]
```

Debidamente combinados los identificadores de una métrica y de los tags, se forman las series temporales que quedan identificadas por ese conjunto de códigos que se denominan TSUID.

2.8. Metadatos.

Los metadatos de OpenTSDB ofrecen información sobre los «data points» almacenados. Los diferentes tipos de metadatos utilizados dentro de OpenTSDB, se describen en los siguientes subapartados.

2.8.1. UIDMeta.

Como ya se ha descrito en apartados anteriores, cada «data point» almacenado tiene al menos tres UUIDs asociados con él. Por un lado tiene el UUID asociado a la métrica y además los «tagk» y «tagv» asociados a cada tag. Cada uno de estos UUID's tienen una entrada en la tabla «tsdb-uid», donde además se registran datos como : uid, type, name y fecha de creación. Otros campos adicionales pueden ser: description, notes, displayName y un conjunto de pares clave-valor que aportan información extra.

2.8.2. TSMeta.

Cada serie temporal queda inequívocamente definida por la combinación de identificadores asociados a su métrica, y a cada par clave-valor que contiene. Estos datos también se almacenan en la tabla «tsdb-uid» y los metaobjetos que lo integran son datos como: tsuid, metric, tags, lastReceived y tiempo de creación y campos adicionales pueden ser: description y notes.

2.8.3. Annotations.

Una anotación es un objeto asociado a un «timestamp» y opcionalmente a una serie temporal. Las anotaciones deben contener una breve descripción limitada a 25 caracteres

2.9. Fechas y Tiempos en OpenTSDB.

Con la finalidad de poder entender adecuadamente los «queries» que se van a poder instrumentar en OpenTSDB y que se desarrollarán en un apartado posterior, es necesario conocer cómo trabaja OpenTSDB con las fechas y los tiempos.

2. Estructura de OpenTSDB.

Existen dos formatos básicos para el tratamiento de la fechas y los tiempos en OpenTSDB: el relativo y el absoluto.

2.9.1. Fechas y tiempos relativos.

Cuando no se conoce el origen de los tiempos para efectuar un cierto query, lo mejor es utilizar este tipo de formato, que básicamente consiste en decir que se obtengan una serie de datos tomados desde el momento actual y la cantidad de tiempo ya pasado que se indique con este tipo de medida de tiempos. El formato de este tipo de indicación de tiempo relativo presenta el siguiente formato:

<cantidad><unidad de tiempo>-ago

Según este formato «<unidad de tiempo>» indica las unidades de tiempo en el que se quiere trabajar (es decir horas, minutos, días, etc) y «<cantidad>» es el número de unidades de tiempo que se quieren elegir. De esta manera si se indica el tiempo de la siguiente manera «2h-ago» se estará haciendo referencia a tomar las medidas desde 2 horas antes hasta el momento actual. Posibles unidades de tiempo serían las siguientes:

- ms -milisegundos.
- s - segundos.
- m - minutos.
- h - horas.
- d - Días (24 horas)
- w -Semanas(7 días)
- n - Meses (30 días)
- y - Años (365 días)

2.9.2. Fechas y tiempos absolutos.

Con este formato se indicará un tiempo de inicio y final de los datos recogidos en formato UNIX, es decir mediante al número de segundos (o milisegundos) transcurridos desde el 1 de enero de 1970 hasta el momento que se indique en la fecha-tiempo indicado (este formato también se denomina epoch).

Las queries lanzadas sobre OpenTSDB también admiten la medición de tiempos en milisegundos. Cuando el tiempo se mide en segundos, el entero facilitado deberá tener un total de 10 dígitos. Si se tuvieran 13 dígitos entonces se estaría trabajando en milisegundos. En el caso de trabajar en milisegundos, el formato del tiempo también se puede dar con un numero decimal, de tal manera que la parte entera tendrá 10 dígitos y la parte decimal 3 dígitos.

A la hora de escribir una query, los datos del inicio y del final de los tiempos, se pueden dar en un formato legible por el ser humano, y los formatos permitidos serían los siguiente:

2. Estructura de OpenTSDB.

- yyyy/MM/dd-HH:mm:ss
- yyyy/MM/dd HH:mm:ss
- yyyy/MM/dd-HH:mm
- yyyy/MM/dd HH:mm
- yyyy/MM/dd

«yyyy» representa al año a 4 dígitos. «MM» indica el mes empezando en 01 para enero y terminando en 12 para el mes de diciembre. «dd» representa el día del mes empezando por 01. «HH» representa la hora del día en formato de 24 horas empezando en 00 hasta llegar a las 23 horas. «mm» representan los minutos empezando en 00 y terminando en 59. «ss» serían los segundos y el valor varía en la misma escala que los segundos.

https://bigdatafran.github.io/big_data/

3. Trabajando con los datos de OpenTSDB.

Existen tres formas de operar con los datos contenidos en OpenTSDB :

1. Utilizando las herramientas que nos proporciona la ventana de comando (CLI).
2. Mediante telnet.
3. Mediante HTTP API.

En los apartados que siguen se pasan a explicar cómo operar en cada uno de los ambientes descritos anteriormente.

3.1. Herramientas CLI.

En apartados anteriores ya se han utilizado comandos de este tipo. Recordemos uno de ellos:

```
./build/tsdb tsd --config=/root/opentsdb/src/opentsdb.conf
```

Que servía para arrancar el servidor OpenTSDB. Otro comando servía para importar datos a la base de datos, recordemos que su sintaxis era la siguiente:

```
/root/opentsdb/build/tsdb import opentsd.input --zkbasedir=/hbase-unsecure --zkquorum
```

En ambos casos lo que se hace es llamar a un fichero ejecutable de Linux (en este caso «tsdb») y después pesarle unos parámetros para indicar lo que queremos que haga. Esa es la estructura general de los comandos CLI proporcionados por OpenTSDB, el cual cuenta dentro de su estructura de ficheros con un solo fichero JAR que es el que recibe los parámetros y actúa en consecuencia.

En los dos comandos anteriores, hemos visto cómo pasando el parámetro «tsd» arranca el servidor, y con el parámetro «import» lo que se consigue es cargar datos en la base de datos. La relación de comandos que proporciona OpenTSDB se detallan en subapartados posteriores.

La forma de ejecutar estos comandos es posicionarse en la carpeta raíz de OpenTSDB y puesto que por defecto el fichero «tsdb» está localizado en el directorio «build» lo que se debe escribir es el comando «./build/tsdb» seguido del comando que se quiere ejecutar.

Los parámetros comunes que suelen acompañar y complementar a los comandos de OpenTSDB, son los que a continuación se pasan a detallar:

3. Trabajando con los datos de OpenTSDB.

```
#!/bin/bash
MY_ARGS='--zkquorum=zookeeper'
set x $MY_ARGS "$@"
shift
```

Figura 3.1.: Contenido fichero tsdb.local

- **--config**. Para indicar el camino completo donde se encuentra el fichero de configuración «opentdb.conf». **Tipo dato**:String; **Ejemplo**: --config=/root/opentsdb/src/opentsdb.conf.
- **--table**. Nombre de la tabla de HBase donde los datos van a ser almacenados. **Tipo dato**:String; **Default**:tsdb; **Ejemplo**: --table=prod-tsdb.
- **--uidtable**.Nombre de la tabla de HBase donde la información de los UID (ver [2.7 en la página 25](#)) va a ser almacenada. **Tipo dato**:String; **Default**:tsdb-uid; **Ejemplo**: --uidtable=prod-tsdb-uid.
- **--verbose**. Para algunas herramientas CLI, este parámetro permite mostrar información sobre los datos gestionados.**Tipo dato**:Boolean.
- **--zkbasedir**. Camino donde se encuentra el znode para la región -ROOT-. **Tipo dato**:String; **Default**:/hbase; **Ejemplo**:--zkbasedir=/prod/hbase.
- **--read-only**. Establecer modo lectura o no de OpenTSDB. **Tipo dato**:Boolean; **Default**:false; **Ejemplo**:--read-only.
- **--zkquorum**. Establecer la lista de servidores y/o puertos de ZooKeeper. **Tipo dato**:String; **Default**:localhost.

La utilización de los parámetros indicados anteriormente puede ser bastante tedioso, sobre todo se de una forma insistentes se están utilizando una y otra vez el mismo conjunto de los mismos. Los creadores de OpenTSDB, han pensado en esto y para facilitar la tarea de los usuarios han diseñado la estrategia que consiste en crear un fichero denominado «./tsdb.local» donde se colocarán los parámetros que se usan con frecuencia y de esta forma evitar tener que teclearlos de una forma continuada. Por ejemplo si de una forma continuada se utiliza «--zkquorum=zookeeper», se puede crear el fichero comentado anteriormente con el contenido que se muestra en la figura

Otra utilidad de este fichero es para determinar la «timezone» en la que se trabaje. En efecto los servidores utilizan normalmente UTC como su «timezone». Por defecto los TSD utilizan la información local del «timezone» para realizar los gráficos. Se puede sobrescribir este valor del «timezone» utilizando la información contenida en el fichero «./tsdb.local». Para hacer esto, si por ejemplo se está en California, se daría información de esta «timezone» utilizando el siguiente comando:

```
echo export TZ=PST8PDT >> tsdb.local
```

3. Trabajando con los datos de OpenTSDB.

```
[root@sandbox opentsdb]# ./build/tsdb uid metrics open
metrics open: [0, 0, 1]
[root@sandbox opentsdb]# ./build/tsdb uid tagk symbol
tagk symbol: [0, 0, 1]
[root@sandbox opentsdb]# ./build/tsdb uid tagv GOOG
tagv GOOG: [0, 0, 2]
[root@sandbox opentsdb]#
```

Figura 3.2.: Comando uid básicos

```
[root@sandbox opentsdb]# ./build/tsdb uid grep metrics 'o*'
metrics : [0, 0, 0, 0, 0, 0, 0, 5]
metrics close: [0, 0, 2]
metrics high: [0, 0, 3]
metrics low: [0, 0, 4]
metrics open: [0, 0, 1]
metrics volume: [0, 0, 5]
```

Figura 3.3.: Ejemplo de uso subcomando «grep».

3.1.1. uid.

Este comando ofrece una serie de utilidades que permiten localizar o modificar información sobre datos contenidos en la tabla «opentsdb-uid». Esta información puede ser relativa a los UID's que se han asignado a las métricas, a los tags relativos a las claves o los tags relativos a los valores. El formato general de este comando es el siguiente:

`uid <subcomandos> [argumentos]`

La utilización básica de este comando se muestra en la figura 3.2 en la que se puede ver cómo sacar información de los uid's que se han asignado en el ejemplo mostrado en el apartado 1.3 en la página 16. Como puede verse, se utiliza la expresión «metrics» para obtener información de una determinada métrica, «tagk» o «tagv» para información del tag-clave o tag-valor respectivamente.

Existen otra serie de parámetros específicos que se utilizan con este comando y que a continuación se pasan a detallar.

3.1.1.1. grep.

Este sub-comando permite utilizar expresiones regulares para localizar UID's. Devuelve los nombre de las expresiones que se adaptan a la expresión regular utilizada, junto con los uid's asignados por el sistema OpenTSB. El formato de este sub-comando es el siguiente:

`grep <tipo> '<expresión regular>'`

Donde «<tipo>» puede tomar los valores de «metrics», «tagk» o «tagv» y a continuación deberá existir una **expresión regular encerrada en comillas simples**.

En la figura 3.3, se puede ver un ejemplo práctico de cómo utilizar este subcomando.

```
[root@sandbox opentsdb]# ./build/tsdb uid assign metrics com.d0 com.d1 com.d2
metrics com.d0: [0, 0, 6]
metrics com.d1: [0, 0, 7]
metrics com.d2: [0, 0, 8]
[root@sandbox opentsdb]#
```

Figura 3.4.: Ejemplo de uso subcomando «assign».

3.1.1.2. assign.

Este sub-comando se utiliza para para asignar IDs a nombres nuevos de métricas, tag-clave o tag-valor. Después de ser ejecutada, devuelve una lista de los valores asignados mediante la ejecución de este subcomando.

El formato que se utiliza es el siguiente:

```
assign <tipo> <name> [<name> ...]
```

Donde «tipo» puede valer lo mismo que se ha indicado en el anterior subapartado y «name» serían de tipo string y serían los nuevos nombres que se quieren añadir a la tabla «tsdb-uid». La figura 3.4 muestra un ejemplo real del uso de este sub-comando.

3.1.1.3. rename.

Con este sub-comando se cambia el nombre de un elemento que ya tiene asignado el UID. Si el nombre que se quiere cambiar no existe, se producirá un error.

Hay que tener en cuenta que OpenTSDB trabaja con la cache, por lo tanto para ver el cambio de inmediato, habrá que vaciar la cache con la siguiente instrucción:

```
curl http://localhost:8091/api/dropcaches
```

Respondiendo el sistema con el siguiente mensaje:

```
{"message": "Caches dropped", "status": "200"}
```

Otra forma de vaciar la caché sería haciendo un «reset» de todos TSDs.

El formato de este sub-comando sería el siguiente:

```
rename <tipo> <name> <newname>
```

Con significado para «tipo» el mismo que se ha indicado en apartados anteriores, «name» sería el nombre que se quiere cambiar y «newname» sería el nuevo nombre que se quiere asignar. Un ejemplo real de este parámetro puede ser el siguiente:

```
./build/tsdb uid rename metrics com.d2 com.d2.new
```

3.1.1.4. delete.

Como perfectamente se puede desprender del nombre asignado a este sub-comando, el mismo se utiliza para borrar de la tabla «tsdb-uid» nombres ya asignados para las métricas, tags-nombre o tags-valor. Como en el apartado anterior para ver el cambio

3. Trabajando con los datos de OpenTSDB.

realizado, se deberá vaciar la caché o hacer un «reset» de todos los TSDs. Tener presente que con este sistema, se borran los UUIDs correspondientes, pero no los datos asociados a esos UUIDs por lo que se pueden crear inconsistencias y generar errores.

El formato de este comando es el siguiente:

```
delete <tipo> <nombre>
```

El significado de los parámetros es el mismo que el que ya se ha explicado en apartados anteriores. Un ejemplo de su uso se muestra a continuación:

```
./build/tsdb ui delete metrics com.d0
```

3.1.1.5. fsck.

Con este sub-comando lo que se hace es escanear la tabla de «opentsdb-uid» en busca de posibles errores. Los errores que encuentra los devuelve y su literal y significado se puede encontrar en la documentación existente en internet. En concreto [haciendo clic en este enlace](#) se puede encontrar la tabla con los posibles errores así como el significado de los mismos.

La estructura de este sub-comando es la siguiente:

```
fsck [fix] [delete_unknown]
```

Con el siguiente significado para los parámetros mostrados:

- **fix**. Intenta corregir los errores que se encuentren.
- **delete_unknown**. Elimina cualquier columna de la tabla UID que no pertenezca a OpenTSDB.

Un ejemplo concreto de utilización de este comando sería el siguiente:

```
./build/tsdb uid fsck fix
```

3.1.1.6. metasync.

Este sub-comando se utiliza para detectar y reparar posibles inconsistencias que pudieran existir en el sistema OpenTSDB. Un ejemplo del uso de este comando se puede ver a continuación:

```
./build/tsdb uid metasync
```

3.1.1.7. metapurge.

Este sub-comando marcará todos los objetos TSMeta y UIDMeta para su eliminación en la tabla UUIDs. Un ejemplo de uso del mismo se puede ver a continuación:

```
./build/tsdb uid metapurge
```

3. Trabajando con los datos de OpenTSDB.

```
[root@sandbox opentsdb]# ./build/tsdb mkmetric sys.user0 sys.user1 sys.user2
metrics sys.user0: [0, 0, 9]
metrics sys.user1: [0, 0, 10]
metrics sys.user2: [0, 0, 11]
[root@sandbox opentsdb]#
```

Figura 3.5.: Ejemplo uso de mkmetric

3.1.1.8. treesync.

Ejecuta la lista de objetos TSMeta en la tabla UID y procesa cada uno a través de todos los árboles configurados y habilitados para compilar ramas. Este sub-comando se puede ejecutar en cualquier momento ya que no tiene efecto sobre los objetos existentes.

Un ejemplo de su uso sería utilizando la siguiente expresión:

```
./build/tsdb uid treesync
```

3.1.1.9. treepurge.

Elimina todas las ramas, la colisión, los datos no coincidentes y opcionalmente, la definición del árbol para un determinado árbol. El formato de este sub-comando es el siguiente:

```
treepurge <id> [definition]
```

Un ejemplo concreto del mismo se muestra a continuación:

```
./build/tsdb uid treepurge 1
```

3.1.2. mkmetric.

Este comando es un atajo de la expresión «uid assign metrics <metric>» (ver apartado anterior). Por lo tanto se utilizará para añadir nombres de métricas nuevas a la tabla «opentsd-uid».

El formato utilizado para este comando es el siguiente:

```
mkmetric metric [metrics]
```

Un ejemplo del mismo, junto a su resultado se puede ver en la figura 3.5.

3.1.3. import.

Este parámetro ya se ha utilizado en un apartado anterior, en concreto en 1.3 en la página 16, y por lo tanto será útil para cargar datos en la base de datos de OpenTSDB. Si el fichero a cargar tiene un volumen de datos elevados, se puede hacer la importación de un fichero comprimido con GZip, y que tenga por tanto una extensión «.gz».

Para que los datos importados se carguen de forma correcta en la base de datos, deberán tener una determinada estructura que debe respetar el formato que se indica a continuación:

```
<métrica> <tiempo> <valor> <tagk=tagv> [... <tagN=tagvN>]
```

3.1.4. query.

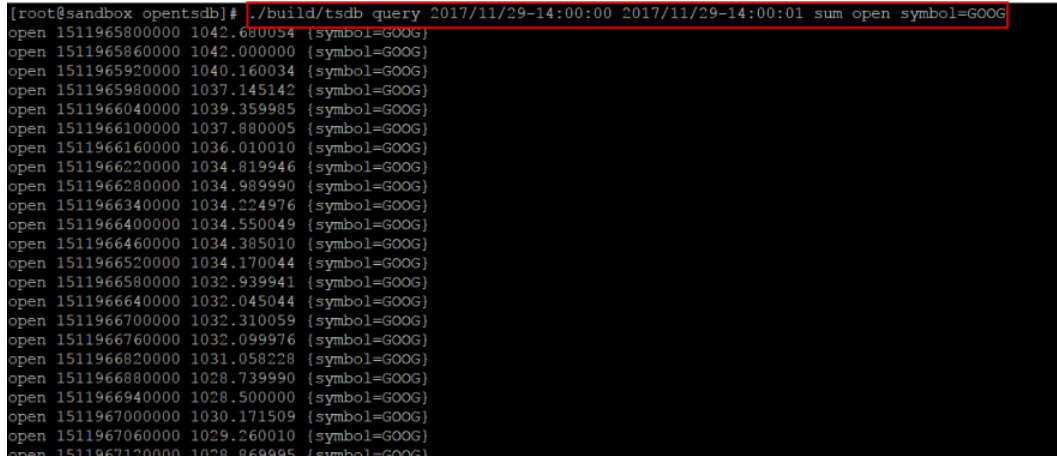
Con este se van a poder sacar series temporales de la base de datos OpenTSDB. El resultado que se muestra es un «data point» por línea. El formato que se debe utilizar es el siguiente:

```
query [Gnuplot opts] START-DATE [END-DATE] <aggregator> [rate] [
  counter,max,reset] [downsample N FUNC] <metric> [<tagk=tagv>]
  [...<tagk=tagv>] [...queries]
```

El significado de los parámetros es el siguiente:

- **Gnuplot opts.** Es un valor opcional utilizado para generar scripts de Gnuplot y gráficos. Es preciso matizar que el gráfico de tipo PNG no se generará, sólo se escribirá el fichero correspondiente que se depositará en el directorio «/tmp». **Tipo dato:** Strings; **Ejemplo:** +wxh=1286x830.
- **START_DATE.** Comienzo de los tiempos para el query a ejecutar. El valor puede estar en formato absoluto o relativo (ver [2.9 en la página 26](#)). **Tipo dato:** String ó entero; **Ejemplo:** 1h-ago.
- **END-DATE.** Es un valor opcional para indicar el final de la toma de datos para el query a lanzar. Si no se da ningún valor se toma el momento de ejecutar la query. Se puede utilizar formato absoluto o relativo (ver [2.9 en la página 26](#)). **Tipo dato:** String ó entero; **Valor defecto:** momento actual; **Ejemplo:** 2016/03/04-01:02:00.
- **aggregator.** Cuando en el resultado de una query se deben agregar valores de diferentes «data-point» se utiliza esta función para obtener el resultado único. **Tipo dato:** String; **Ejemplo:** sum.
- **rate.** . **Tipo dato:** String; **Ejemplo:** rate.
- **counter.** Literal que es opcional, . **Tipo dato:** String; **Ejemplo:** counter.
- **max.** Es un entero positivo que representa el máximo valor que puede tomar el contador. **Tipo dato:** Integer; **Valor defecto:** java Long.MaxValue; **Ejemplo:** 60500.
- **resetValue.** Es un valor opcional, de tal manera que si se indica un valor para este parámetro, cuando se excede el «aggregator» vuelve a cero en lugar de calcular la tasa correspondiente. Es útil cuando las fuentes de datos se restablecen con frecuencia para evitar picos «espurios». **Tipo dato:** Entero; **Ejemplo:** 60000.
- **downsample N FUNC.** . **Tipo dato:** String; **Ejemplo:** downsample 200000 avg.
- **metric.** Es un valor obligatorio ya que hay que indicar el nombre de la métrica sobre la que se quiere obtener resultados. **Tipo dato:** String; **Ejemplo:** dat.c0.
- **tagk=tagv.** Opcional conjunto de pares tag-nombre y tag-valor para precisar los datos que se quieren obtener. **Tipo dato:** String; **Ejemplo:** symbol=GOOG.

3. Trabajando con los datos de OpenTSDB.



```
[root@sandbox opentsdb]# ./build/tsdb query 2017/11/29-14:00:00 2017/11/29-14:00:01 sum open symbol=GOOG
open 1511965800000 1042.680054 {symbol=GOOG}
open 1511965860000 1042.000000 {symbol=GOOG}
open 1511965920000 1040.160034 {symbol=GOOG}
open 1511965980000 1037.145142 {symbol=GOOG}
open 1511966040000 1039.359985 {symbol=GOOG}
open 1511966100000 1037.880005 {symbol=GOOG}
open 1511966160000 1036.010010 {symbol=GOOG}
open 1511966220000 1034.819946 {symbol=GOOG}
open 1511966280000 1034.989990 {symbol=GOOG}
open 1511966340000 1034.224976 {symbol=GOOG}
open 1511966400000 1034.550049 {symbol=GOOG}
open 1511966460000 1034.385010 {symbol=GOOG}
open 1511966520000 1034.170044 {symbol=GOOG}
open 1511966580000 1032.939941 {symbol=GOOG}
open 1511966640000 1032.045044 {symbol=GOOG}
open 1511966700000 1032.310059 {symbol=GOOG}
open 1511966760000 1032.099976 {symbol=GOOG}
open 1511966820000 1031.058228 {symbol=GOOG}
open 1511966880000 1028.739990 {symbol=GOOG}
open 1511966940000 1028.500000 {symbol=GOOG}
open 1511967000000 1030.171509 {symbol=GOOG}
open 1511967060000 1029.260010 {symbol=GOOG}
open 1511967120000 1028.866995 {symbol=GOOG}
```

Figura 3.6.: Ejemplo de una query en la ventana de comandos

- **Querys adicionales.** Parámetro opcional con indicación de más querys a ejecutar. Cada query debe tener el mismo formato de tiempo de comienzo con un «agregador». Todas las querys deben tener el mismo comienzo de los tiempos. **Tipo dato:** String; **Ejemplo:** sum tsd.habse=hdp.

En la figura 3.6, se muestra una query para obtener información de los datos cargados en OpenTSDB tal y como se explicó en el apartado 1.3 en la página 16. Como puede verse el formato de salida de la query es el siguiente:

<metrica> <tiempo> <valor> {<tagk=tagv> [...<tagkN=tagvN>]}

3.1.5. fsck.

3.1.6. scan.

3.1.7. search.

3.1.8. tsd.

3.2. Telnet API

Lo primero que hay que hacer en este caso es conectarse con OpenTSDB vía telnet. Para ello el sistema que debe utilizar es el que se muestra en la figura 3.7¹:

Una vez se tenga la conexión establecida vía telnet, los parámetros que admite son los siguientes:

¹Hay que tener en cuenta que la conexión así establecida se cierra de un determinado periodo sin actividad que normalmente son los 5 minutos. Para cerrar la conexión, se utilizará el comando «exit»

3. Trabajando con los datos de OpenTSDB.

```
[root@sandbox ~]# telnet localhost 8091
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
Trying ::1...
telnet: connect to address ::1: Cannot assign requested address
[root@sandbox ~]#
```

Figura 3.7.: Inicio de telnet

3.2.1. put.

Con este comando se podrá insertar información en la base de datos de OpenTSDB. El formato para utilizarlo sería el siguiente:

```
put <metric> <timestamp> <value> <tagk_1>=<tagv_1>[ <tagk_n>=<tagv_n>]
```

Un ejemplo de uso de este comando en el ejemplo utilizado sería el siguiente:

```
put open 0000234567 34.5 symbol=GOOG
```

Ejemplos de errores obtenidos mediante uso inadecuado de este comando se puede ver en los siguientes comandos:

```
put
put: illegal argument: not enough arguments (need least 4, got 0)
put open 0000234567 35.2 symbol=GOOG2
unknown command: put. Try 'help'.
```

3.2.2. rollup.

3.2.3. histogram.

3.2.4. stats.

Con este comando lo que se consigue es una lista de estados de TSD, uno estado en cada línea. El formato de este comando es bien simple:

```
stats
```

3.2.5. version.

Con este comando se puede obtener la versión de OpenTSDB con la que se esté trabajando. El formato de este comando es bien sencillo.

```
version
```

3.2.6. help.

Nos ofrece una lista de los comandos que se pueden utilizar. Su formato es el siguiente:

```
help
```

3. Trabajando con los datos de OpenTSDB.

Y la salida que muestra es la siguiente:

```
available : put dropcaches version exit help diediedie
```

3.2.7. dropcaches.

Purga las caches de la métrica, del tag-clave y tag-value. La sintaxis es la siguiente:

```
dropcaches
```

La salida que muestra es la siguiente:

```
Caches dropped
```

3.2.8. diediedie.

Con este comando apaga el proceso TSD y se sale de telnet.

3.3. HTTP API.

Primero decir, que el API HTTP es de tipo RESTful. El intercambio de datos utilizado por defecto es vía JSON, aunque se pueden usar otro tipo de formatos.

²

Este API utiliza una serie de palabras clave, denominadas Verbos, que puede tomar los valores que a continuación se detallan:

Verb	Descripción	Override
GET	Se suele utilizar para recuperar datos desde OpenTSDB. Request vía GET sólo puede utilizar «query string parameters»	N/A
POST	Para actualizar o crear un objeto en OpenTSDB, utilizando el contenido del cuerpo para hacer la «request»	method_override= =post
PUT	Para reemplazar por completo un determinado objeto	method_override= =put
DELETE	Utilizado para borrar datos	method_override= =delete

Sin embargo hay que tener en cuenta, que Verbos como DELETE o PUT pueden ser bloqueados por los cortafuegos o los proxis. Por ese motivo muchas peticiones pueden ser hechas con el verbo GET pero añadiendo el parámetro «method_override». Con este parámetro se pueden pasar los datos en formato de «query-string» en lugar de utilizar «body content». Para ver cómo utilizarlo, a continuación se pasa un ejemplo de cómo poder borrar un determinado «data point»:

```
/api/annotation?start_time=343434&tsuid=20&method_override=delete
```

²REST (Representation State Transfer) es una arquitectura que se ejecuta sobre HTTP.

RESTful hace referencia a un servicio web que implementa la arquitectura REST

3. Trabajando con los datos de OpenTSDB.

TS#	Métrica	Tags	Valores
1	metrica	tag1=a tag2=b	2
2	metrica	tag1=a tag2=c	5
3	metrica	tag1=a tag2=d	1
4	metrica	tag2=b	3
5	metrica	tag2=b propi=fran	7
6	metrica	tag1=d tag2=b	4
7	metrica	tag1=d tag2=c	3
8	metrica	tag1=d	6

Cuadro 3.1.: Tabla de datos ejemplo

3.3.1. Filtros Query.³

Con estas herramientas lo que se consigue es filtrar los datos para obtener subconjuntos de datos más pequeños con los que poder operar y trabajar de na forma más eficiente.

Para entender mucho mejor el funcionamiento de este sistema, se mostrará en la tabla 3.1 una serie de datos, sobre los que se van a aplicar los filtros que se detallarán después para de esta manera mejorar la exposición de los conceptos.

3.3.1.1. Agrupación de datos.

Este proceso de agregación consiste en combinar múltiples series temporales en una sola utilizando para ello una función de agrupación y filtros. Por defecto, OpenTSDB agrupa por el métricas, y por lo tanto si una query devolviera por ejemplo 6 series temporales con una función de agrupación igual a «sum», estas 6 series se agruparían para formar una sola mediante la suma de los valores de las mismas.

Si no se quisiera agrupar los datos y se desea obtener individualmente cada serie temporal, desde la versión 2.2 de OpenTSDB se ha incluido la función de agregación «none» que permite conseguir este objetivo.

Inicialmente, y hasta la versión 2.1 de OpenTSDB existían sólo dos tipos de operadores de filtros. Estos dos tipos de operadores eran los siguientes:

- *.- El asterisco (también conocido en el argot informático como wildcard), gracias al cual se devuelven resultados separados por cada tag-clave que se detecte.
- |.- También denominado pipe (o literal_or) . Actuaría de forma similar al operador OR muy utilizado en el campo de la informática. Más adelante se pondrán ejemplos que aclararán mucho mejor estos conceptos.

A continuación se muestran algunos ejemplos ⁴.

³Para ver los filtros disponibles en OpenTSDB 2.2 en adelante, se puede utilizar en la barra de direcciones del navegador la expresión <http://localhost:8091/api/config/filters>

⁴Hay que aclarar que con la opción «m» se va a hacer referencia a la función de agregación. Después dos puntos seguidos de la métrica y entre llaves los pares de valores correspondientes a los tags que se desean filtrar

3. Trabajando con los datos de OpenTSDB.

Ejemplo 1: `http://localhost:8091/q?start=2h-ago&m=sum:metrica{tag2=b}`

El resultado que se obtiene es el siguiente:

Series incluidas	Tags	Valor obtenido
1,4,5,6	tag2=b	16

Ejemplo 2: `http://localhost:8091/q?start=3h-ago&m=sum:metrica{tag1=a, tag2=b}`

Series incluidas	Tags	Valor obtenido
1	tag1=a tag2=b	2

Ejemplo 3: `http://localhost:8091/q?start=2h-ago&m=sum:metrica{tag2=*, tag1=a}`

Series incluidas	Tags	Valor obtenido
1,2,3,4,5,6,7	tag1=a tag2=* (cualquier valor)	25

Ejemplo 4: `http://localhost:8091/q?start=2h-ago&m=sum:metrica{tag2=b|c}`

Series incluidas	Tags	Valor obtenido
1,2,4,5,6	tag2=b ó c	21

Explicit Tags.

Esta facilidad es nueva desde la versión 2.3 de OpenTSDB y si se conocen todos los tags-key de una métrica, se puede mejorar considerablemente la velocidad de la consulta utilizando este factor añadido. Los ejemplos que a continuación de muestran, tienen el parámetro m para definir la función de agregación, le sigue los dobles puntos y después la expresión «explicit_tags» junto a dos puntos a la derecha más el nombre de la métrica y entre llaves los pares clave-valor de los tags a consultar.

Ejemplo 4: `http://localhost:8091/q?start=1h-ago&m=sum:explicit_tags:metrica{tag2=b}`

Series incluidas	Tags	Valor obtenido
4	tag2=b	3

3.3.1.2. Formato Query de una métrica.

La especificación completa de este tipo de queries es como a continuación de expone:

```
m=<aggregator>:[rate[{counter[,<counter_max>[,<reset_value>]]]:}
[<down_sampler>:][percentiles\(<p1>,<pn>\):][explicit_tags:]
<metric_name>[{<tag_name1>=<grouping_filter>
[,...<tag_nameN>=<grouping_filter>]}][{<tag_name1>=<non_grouping
filter>
[,...<tag_nameN>=<non_grouping_filter>}]}
```

Una forma más fácil de construir este tipo de queries es utilizar el GUI que ofrece OpenTSDB. Si se usa este elemento de representación gráfica utilizando los campos que contiene, se puede construir la consulta que se desee y entonces en la barra de direcciones del navegador que se esté utilizando, se podrá ver la query que se ha obtenido. Cuando se tenga esto, se puede copiar y pegar la query y adaptarla a las circunstancias concretas que se necesiten.

3.3.1.3. Formato Query de un TSUID.

Este tipo de queries son más simples que las mostradas anteriormente, simplemente hay que pasar una lista de uno o más códigos hexadecimales asociados a los TSUIDs separados por comas. Su especificación se muestra a continuación:

```
tsuid=<aggregator>:<tsuid1>[,...<tsuidN>]
```

Un par de ejemplo de queries, se muestran a continuación.

```
http://localhost:8091/api/query?start=1h-ago&m=sum:rate:proc.stat.cpu  
{host=foo,type=idle}  
http://localhost:8091/api/query?start=1h-ago&tsuid=sum  
:000001000002000042,000001000002000043
```

3.3.2. API Endpoints.

A continuación se pasan a explicar los códigos APIS que se pueden utilizar para trabajar con OpenTSDB.

3.3.2.1. /s

Con este código se va a tener acceso a los ficheros estáticos de OpenTSDB. Recordar que el path a estos ficheros estáticos se define en el fichero de configuración dando un valor al parámetro «tsd.http.staticroot», o si se utiliza un parámetro den CLI mediante la propiedad «-- staticroot».

En el fichero de configuración que se utilizado, el camino dado para esta carpeta es el siguiente: «/root/opentsdb/build/staticroot», si miramos en su contenido, uno de los ficheros que contiene es «queryui.nocache.js». Si en la barra de direcciones de un navegador se pone «http://localhost:8091/s/queryui.nocache.js», se podrá ver el contenido de este fichero javascript.

3.3.2.2. /api/aggregators.

Este código sirve para obtener las funciones de agregado que están implementadas para la utilización en las queries.

Los verbos con los que se pueden utilizar son : GET y POST.

Para su utilización se puede hacer de las siguientes maneras.

- En la líneas de comandos poner el siguiente: «curl http://localhost:8091/api/aggregators». La salida que se obtiene se puede ver en la figura 3.8.
- Poniendo en la barra de direcciones de un navegador la instrucción «http://localhost:8091/api/aggregators». La salida que se obtiene se puede ver en la figura 3.9.

3. Trabajando con los datos de OpenTSDB.

```
Last login: Mon Nov 13 14:53:06 2017 from 10.0.2.2
[root@sandbox ~]# curl http://localhost:8091/api/aggregators
[["mult","p90","zimsum","mimmax","sum","p50","none","p95","ep99r7","p75","p99","ep99r3","ep95r7","min","avg","ep75r7","dev","ep95r3","ep75r3","ep50r7","ep90r7","mimmin","p999","ep50r3","ep90r3","ep999r7","last","max","count","ep999r3","first"]]
[root@sandbox ~]# curl http://localhost:8091/api/aggregators
```

Figura 3.8.: Salida aggregators utilizando curl.



Figura 3.9.: Salida aggregator con navegador.

3.3.2.3. /api/annotation.

Con este código, se va a poder añadir, Time»editar o borrar cualquier tipo de anotación sobre la base de datos. Estas anotaciones constituyen buenas herramientas para poder recordar hechos y acontecimientos en el momento de la generación de los «data points».

Con este código se puede hacer una sola anotación. Para hacer más de una anotación al mismo tiempo, se deberá utilizar «/api/annotation/bulk».

Los verbos que se pueden utilizar con este código son los siguientes:

- GET. Para generar una sola anotación.
- POST. Para vcrear o modificar una anotación.
- PUT. Crea o reemplaza una anotación.
- DELTE. Para borrar una anotación.

Las anotaciones quedan caracterizadas por el campo «startTime» y con carácter opcional con el otro campo denominado «tsuid». Cada anotación que se haga, puede ser de **tipo global** (estando por tanto asociada a todas las series), o puede ser de **tipo local**, en cuyo caso estará asociado a la serie identificada por el parámetro «tsuid» indicado. Si no se facilita el parámetro «tsuid» o tiene un valor vacío, entonces la anotación se considerará que es de tipo global. Los parámetros soportados son los siguientes:

- **startTime**. Un dato de tipo epoch Unix, en segundos. **Tipo dato:**Integer;**Requerido:**Si; **Query-string:**start_time; **Ejemplo:**2345643764.
- **endTime**. Un dato de tipo epoch Unix, indicando final del tiempo:**Tipo dato:**Integer;**Requerido:**No; **Query-string:**end_time; **Ejemplo:**2345643777.
- **tsuid**. Un valor de un TSUID si la anotación va a estar asociada con una serie.**Tipo dato:**String;**Requerido:**No; **Query-string:**tsuid; **Ejemplo:**000001000001000001.

3. Trabajando con los datos de OpenTSDB.

- **description.** Una breve descripción del evento, lo ideal es que tenga menos de 25 caracteres. **Tipo dato:**String; **Requerido:**No; **Query-string:**description; **Ejemplo:** network machine
- **notes.** Notas detalladas del evento. **Tipo dato:**String; **Requerido:**No; **Query-string:**notes; **Ejemplo:** Cambiado por elemento 4 por desgaste.
- **custom.** Un map de parejas clave-valor para almacenar campos y valores a medida. **Tipo dato:**Map; **Requerido:**No.

Un ejemplo de uso con una petición de tipo GET sería el siguiente:

```
http://localhost:8091/api/annotation?start_time=3465627475&tsuid=000001000001000001
```

En el supuesto de que se haga vía una petición de tipo POST, el comando a utilizar dentro de la línea de comandos del shell, sería el siguiente:

```
curl -X POST --data-binary "@/root/fichero.json" --header "Content-Type: application/json" http://localhost:8091/api/annotation
```

Siendo el contenido del fichero «/root/fichero.json» el siguiente:

```
{
  "startTime": "2374756368",
  "tsuid": "000001000001000001",
  "description": "Una breve descripcion"
}
```

3.3.2.4. /api/annotation/bulk.

Sirve para añadir, modificar o borrar más de una anotación. El tratamiento es similar al que se ha descrito en el anterior apartado, lo único que las anotaciones están contenidas dentro de un array.

3.3.2.5. /api/config.

Con este elemento se obtiene información de la configuración de OpenTSDB. Un ejemplo de su uso es el siguiente:

```
http://localhost:8091/api/config
```

3.3.2.6. /api/config/filters.

(implementado de la versión 2.2 en adelante). Con este comando se pueden obtener los filtros cargados en TSD y alguna información sobre su uso. Un ejemplo de su uso es el siguiente:

```
http://localhost:8091/api/config/filters
```

3.3.2.7. /api/dropcaches.

Con esta instrucción se hace una limpieza del cache de OpenTSDB, esto incluye todos los UUIDs de las métricas, así como los tags-nombre y tags-valor. Un ejemplo de su uso es el siguiente:

```
http://localhost:8091/api/dropcaches
```

3.3.2.8. /api/put.

Esta instrucción se utilizará para enviar datos a OpenTSDB para su almacenamiento vía un protocolo HTTP y puede ser utilizado como una alternativa al protocolo Telnet o a la línea de comandos CLI ya comentado en apartados anteriores. El envío de la información a la base de datos, **sólo se permite sea realizado mediante el protocolo POST**.

Cuando se realiza un envío de la información con varios «data point», el envío se hace uno a uno, de tal manera que los datos erróneos serán rechazados pero los correctos serán aceptados. Es decir, que el hecho de que un «data point» presente algún error no impide seguir procesando la información.

A continuación se muestran los parámetros que se pueden añadir como «query string» y que alteran la respuesta dada.

- **summary**. Para indicar si devuelve o no un resumen de la información enviada. **Tipo dato**: Presente; **Requerido**: opcional; **V. defecto**: false; **Query String**: summary; **Ejemplo**: /api/put?summary .
- **details**. Si devuelve o no información detallada. **Tipo dato**: Presente; **Requerido**: opcional; **V. defecto**: false; **Query String**: details; **Ejemplo**: /api/put?details .
- **sync**. Para indicar si hay que esperar a que los datos se almacenen antes de devolver los resultados. **Tipo dato**: Boolean; **Requerido**: opcional; **V. defecto**: false; **Query String**: sync; **Ejemplo**: /api/put?sync .
- **sync_timeout**. Un valor de tiempo de espera en milisegundos para esperar a que los datos sean almacenados antes de devolver un error. Cuando se pasa de ese tiempo, si se usa el parámetro «details» se dará información de los datos que no han presentado error y los que tienen error. Se debe indicar también el parámetro «sync» para que esta medida tenga efecto.

En el supuesto de que se indiquen los parámetros «detailed» y «summary» al mismo tiempo, el API responderá como si se hubiera indicado solamente «detailed».

A continuación se indican los campos que se pueden incorporar en el fichero JSON que se utilice para enviar la información a la base de datos.

- **metric**. Para indicar el nombre de la métrica a la que se refiere la información. **Tipo dato**: String; **Requerido**: SI.

3. Trabajando con los datos de OpenTSDB.

- **timestamp**. Un tiempo en formato epoch Unix dado en segundos o milisegundos. **Tipo dato**: Entero; **Requerido**: SI.
- **valor**. Para indicar el valor de la medición que se quiere hacer. **Tipo dato**: Entero, Flotante, String; **Requerido**: SI.
- **tags**. Un map conteniendo tags de tipo clave-valor. Al menos debe facilitarse un par. **Tipo dato**: Map; **Requerido**: SI; Ejemplo: { «clave»: «valor» }.

Se ha ejecutado la siguiente sentencia:

```
curl -X POST --data-binary "@/root/fichero.json" --header "Content-Type: application/json" http://localhost:8091/api/put?details=true
```

El fichero json utilizado contiene la siguiente información:

```
{
  "metric": "open",
  "timestamp": 2324234525,
  "value": 13,
  "tags": {
    "type": "GOOG"
  }
}
```

El resultado que ha salido ha sido⁵:

```
{"success": 1, "failed": 0, "errors": []}
```

Como puede comprobarse, se ha grabado el dato que se ha enviado, y ha habido cero errores en el envío.

Un ejemplo de envío de dos «data points» es el que se muestra a continuación, donde se puede comprobar que los datos están encerrados en un array.

```
[
  {
    "metric": "open",
    "timestamp": 5324245253,
    "value": 23,
    "tags": {
      "type": "GOOG"
    }
  },
  {
    "metric": "open",
    "timestamp": 3324349257,
    "value": 73,
    "tags": {
      "type": "GOOG"
    }
  }
]
```

⁵Observar que la salida tanto con el query string «summary» como con «details» es la misma, dado que no hay ningún error. De haber algún error, y de utilizar «details» se obtendría ls detalles del error.

3. Trabajando con los datos de OpenTSDB.

```
}  
]
```

Otro ejemplo puede ser ejecutar la siguiente instrucción:

```
curl -X POST -d "@/root/sample.json" -H "Accept: Application/json" -H  
"Content-Type: application/json" http://localhost:8091/api/put
```

sobre el fichero de tipo json:

```
[  
  {  
    "metric": "open",  
    "timestamp": 1346846401,  
    "value": 42.5,  
    "tags": {  
      "type": "GOOG"  
    }  
  },  
  {  
    "metric": "open",  
    "timestamp": 1346846402,  
    "value": 2.5,  
    "tags": {  
      "type": "GOOG"  
    }  
  },  
  {  
    "metric": "open",  
    "timestamp": 1346846402,  
    "value": 3.5,  
    "tags": {  
      "type": "GOOG"  
    }  
  }  
]
```

Si ahora se pone en la barra de direcciones del navegador lo siguiente:

```
http://localhost:8091/api/query?start=0&m=sum:open{type=GOOG}
```

Se obtiene el siguiente resultado:

```
[{"metric": "open", "tags": {"type": "GOOG"}, "aggregateTags": [], "dps"  
  ": {"2": 30, "8324234": 23, "8324245": 23.0, "1346846401": 42.5, "1346846402": 3.5} }]
```

3.3.2.9. /api/rollup

Con este comando se puede almacenar «rollup» y datos de tipo «pre-aggregated» en OpenTSDB utilizando HTTP (ver apartados anteriores para saber en qué consisten este tipo de datos). El envío se hace siempre con peticiones de tipo POST.

Los parámetros y el significado que soportan son los mismos que los usados para «/api/put» por lo que no se describen de nuevo en este apartado. A esa lista de parámetros hay que añadir tres más que son los siguientes:

- **Interval.** Un intervalo de tiempo, dentro del cual se va a emplear la función de agregación. El intervalo tiene el formato: <cantidad><unidad tiempo> (por ejemplo 2h para indicar 2 horas).
- **aggregator.** Una función de agregación utilizada para calcular el valor de los datos agregados. Un ejemplo puede ser la función SUM.
- **groupByAggregator.** Para indicar una función de agregación para generar los valores pre-agregados.

3.3.2.10. /api/histogram

3.3.2.11. /api/query.

Este comando es uno de los más utilizados, dentro de las instrucciones API. Los tipos de petición que admite son: GET, POST, DELETE. El tipo DELETE se utiliza para el borrado de «data points», pero para poder hacer esto el parámetro del fichero de configuración «tsd.http.query.allow_delete» debe estar activado. Los datos que se borran, van a ser devueltos en la query indicada para el borrado de los datos.

Los parámetros utilizados en la request incluyen los siguientes:

- **start.** El tiempo en el que comienza a contar la query. **Tipo dato:**String,Integer; **Requerido:**SI ; **Query String:**start; **Ejemplo:**1h-ago.
- **end.** Final del tiempo para la ejecución de la query. **Tipo dato:**String,Integer; **Requerido:** NO; **V. defecto:** momento actual; **Query String:**start; **Ejemplo:**1h-ago.
- **queries.** Una o más sub queries utilizadas para seleccionar las series temporales devueltas. Estos pueden referirse a métricas «m» o a TSUID «tsuids» queries. **Tipo dato:**Array; **Requerido:** SI; **Query String:** m ó tsuids.
- **noAnnotations.** Para indicar si devuelve o no annotations con una query. **Tipo dato:**Boolean; **Requerido:** NO; **Query String:** no_annotations.
- **globalAnnotations.** Para indicar si la query devuelve anotaciones de tipo global. **Tipo dato:**Boolean; **Requerido:** NO; **Query String:** global_annotations; **v. defecto:** false.

3. Trabajando con los datos de OpenTSDB.

- **msResolution (6 ms)**. Para indicar si o no los «data-points» obtenidos están en segundos o en milisegundos. **Tipo dato:** Boolean; **Requerido:** NO; **Query String:** ms; **v. defecto:** false.
- **showTSUIDs**. Si se muestran o no en la salida los TSUIDs asociados con la serie temporal obtenida. **Tipo dato:** Boolean; **Requerido:** NO; **Query String:** show_tsuids; **v. defecto:** false.
- **showSummary**. Si se muestra o no un pequeño resumen junto a los resultados. **Tipo dato:** Boolean; **Requerido:** NO; **Query String:** show_summary; **v. defecto:** false.
- **showStats**. Si se muestra o no determinadas estadísticas con los resultados. **Tipo dato:** Boolean; **Requerido:** NO; **Query String:** show_stats; **v. defecto:** false.
- **showQuery**. Si se devuelve o no el subquery original junto a los resultados. **Tipo dato:** Boolean; **Requerido:** NO; **Query String:** show_query; **v. defecto:** false.
- **delete**. Puede ser pasado a el JSON con una petición POST para borrar cualquier «data-point» que verifique lo indicado en la query. **Tipo dato:** Boolean; **Requerido:** NO; **v. defecto:** false.
- **timezone**. Es un valor opcional de la «timezone». **Tipo dato:** String; **Requerido:** NO; **Query String:** timezone; **v. defecto:** UTC.
- **useCalendar**. Para indicar si se usa o no el calendario basado en la «timezone» definida para los intervalos utilizados en los downsampling. **Tipo dato:** String; **Requerido:** NO; **v. defecto:** false.

Los **Sub Queries** son una parte esencial para la generación de los de los queries admitidos en OpenTSDB. Existen dos tipos de Sub Queries:

- **Metric Query**. Con esta opción se suministra el nombre completo de una métrica junto a una lista opcional de tags. Esto está optimizado para poder agregar múltiples series temporales en un sólo resultado.
- **TSUID Query**. Es una lista de uno o más TSUIDs que comparten una métrica común. esto está optimizado para buscar series individuales donde no se requiere ningún tipo de agrupación.

Una query de OpenTSDB puede tener más de un subquery y cualquier mezcla de los dos tipos indicados anteriormente.

En cada uno de los subqueries, se pueden añadir los siguientes campos:

- **aggregator**. Es el nombre de la función de agregación. **Tipo dato:** String; **Requerido:** SI; **Ejemplo:** sum .
- **metric**. El nombre de la métrica almacenada en el sistema. **Tipo dato:** String; **Requerido:** SI; **v. defecto:** false; **Ejemplo:** sum .

3. Trabajando con los datos de OpenTSDB.

- **rate**. Si o no los datos devueltos en una query se convierten en valores de tipo razón. **Tipo dato**: Boolean; **Requerido**: NO; **V. defecto**: false.
- **rateOptions**. Se explica a continuación de la presente lista.
- **downsample**. Una función de tipo downsample para reducir la cantidad de datos devueltos. **Tipo dato**: Boolean; **Requerido**: NO; **Ejemplo**: 5m-avg.
- **tags**. Para clasificar las series temporales por tags. **Tipo dato**: Boolean; **Requerido**: NO.
- **filters**. Para filtrar las series temporales devueltas. **Tipo dato**: List.
- **explicitTags**. Devuelve las series que incluyen los tags-key indicados en los filtros. **Tipo dato**: Boolean; **Requerido**: NO.
- **percentiles**. Obtiene datos del histograma para la métrica y calcula la lista dada de percentiles de los datos. **Tipo dato**: List; **Requerido**: NO; ejemplo: [99.9, 95.0, 75.0].

Rate Options.

Cuando se añaden «rate options» en un query string, las opciones deben estar encerradas entre llaves. Un ejemplo sería el siguiente: `m=sum:rate{counter,,100}:if.octets.in`. Los sub-campos soportados en el campo rateOptions son los siguientes:

- **counter**. Si los datos subyacentes son o no un contador que se incrementa de forma monótona. **Tipo dato**: Boolean; **Requerido**: NO.
- **counterMax**. Un entero positivo que representa el valor máximo de un contador. **Tipo dato**: Integer; **Requerido**: NO; **V. defecto**: Java Long.MaxValue.
- **resetValue**. Representa un valor opcional, que cuando es superado implica que el agregado vuelve a cero en lugar de calcular el valor del rate. **Tipo dato**: Integer; **Requerido**: NO; **V. defecto**: 0.
- **dropResets**. Si se da o no un simple drop rolled-over o se hace un reset de los «data points». **Tipo dato**: Boolean; **Requerido**: NO; **V. defecto**: false.

Downsampling.

Downsampling sirve para agrupar los datos por intervalos temporales de tal forma que los datos obtenidos sean más interpretables que los originales. Su formato es el siguiente:

`<intervalo><unidades><aggregator>[c][-<fill policy>]`

A continuación se muestran algunos ejemplos.

```
1h-sum
20m-avg-nan
15h-max-zero
```

Filtros.

Para más detalle, ver apartado [3.3.1 en la página 39](#).

3.3.2.12. /api/search.

Con esta instrucción lo que se hace es localizar metadatos de OpenTSDB. Recordar que estos datos están recogidos en la tabla «tsdb-meta». Este comando admite peticiones de tipo GET y POST.

Los parámetros admitidos por ese comando son los siguientes:

- **query**. Es el string pasado al motor de búsqueda. **Tipo Dato**: String; **Requerido**:NO; **Query string**:query.
- **limit**. Limita el número de resultados devueltos por la query. **Tipo Dato**: Entero; **Requerido**:NO; **V. defecto**:25; **Query string**:limit.
- **startIndex**. Usado en combinación con «limit» para saltar los primeros resultados. **Tipo Dato**: Entero; **Requerido**:NO; **V. defecto**:0; **Query string**: start_index.
- **metric**. El nombre de una métrica o un wildcard para buscar queries. **Tipo Dato**: String; **Requerido**:NO; **V. defecto**:* ; **Query string**: metric.
- **tags**. Uno o pares de tipo clave-valor (tags) para buscar en query. **Tipo Dato**: Array; **Requerido**:NO; **Query string**: tags.

A continuación se muestra un ejemplo, utilizando un Query String:

`http://localhost:8091/api/search/tsmeta?query=name:*&limit=3&start_index=0`

Y usando una expresión POST, sería lo siguiente:

```
{
  "query": "name:*",
  "limit": 4,
  "startIndex": 5
}
```

3.3.2.13. /api/serializers.

Con este comando se listan los plugins de serialización cargados en TSD. Si en la barra de direcciones del navegador se pone:

`http://localhost:8091/api/serializers`

Se obtendrá una respuesta del tipo que a continuación se muestra:

```
[{"request_content_type":"application/json","formatters":["DropCachesV1","SuggestV1","AggregatorsV1","AnnotationV1","AnnotationsV1","AnnotationBulkDeleteV1","VersionV1","StatsV1","RegionStatsV1","ThreadStatsV1","JVMStatsV1","QueryStatsV1","FilterConfigV1","ConfigV1","SerializersV1","TreesV1","TreeV1","TreeRuleV1","TreeTestV1","SearchResultsV1","BranchV1","UidAssignV1","UidMetaV1","TSMetaV1","TSMetaListV1","UidRenameV1","TreeCollisionNotMatchedV1","ErrorV1","ErrorV1","NotFoundV1","PutV1","LastPointQueryV1","QueryAsyncV1","QueryV1"],"}
```

3. Trabajando con los datos de OpenTSDB.

```
{
  "metric": "tsd.connectionmgr.connections", "timestamp": 1510603063, "value": "1", "tags": {
    "host": "sandbox.hortonworks.com", "type": "open"
  }
},
{
  "metric": "tsd.connectionmgr.connections", "timestamp": 1510603063, "value": "0", "tags": {
    "host": "sandbox.hortonworks.com", "type": "rejected"
  }
},
{
  "metric": "tsd.connectionmgr.connections", "timestamp": 1510603063, "value": "11", "tags": {
    "host": "sandbox.hortonworks.com", "type": "total"
  }
},
{
  "metric": "tsd.connectionmgr.exceptions", "timestamp": 1510603063, "value": "0", "tags": {
    "host": "sandbox.hortonworks.com", "type": "closed"
  }
},
{
  "metric": "tsd.connectionmgr.exceptions", "timestamp": 1510603063, "value": "0", "tags": {
    "host": "sandbox.hortonworks.com", "type": "reset"
  }
},
{
  "metric": "tsd.connectionmgr.exceptions", "timestamp": 1510603063, "value": "0", "tags": {
    "host": "sandbox.hortonworks.com", "type": "timeout"
  }
},
{
  "metric": "tsd.rpc.received", "timestamp": 1510603063, "value": "0", "tags": {
    "host": "sandbox.hortonworks.com", "type": "telnet"
  }
},
{
  "metric": "tsd.rpc.received", "timestamp": 1510603063, "value": "15", "tags": {
    "host": "sandbox.hortonworks.com", "type": "http"
  }
},
{
  "metric": "tsd.rpc.received", "timestamp": 1510603063, "value": "0", "tags": {
    "host": "sandbox.hortonworks.com", "type": "http_plugin"
  }
},
{
  "metric": "tsd.rpc.exceptions", "timestamp": 1510603063, "value": "0", "tags": {
    "host": "sandbox.hortonworks.com"
  }
},
{
  "metric": "tsd.http.latency_50pct", "timestamp": 1510603063, "value": "8", "tags": {
    "host": "sandbox.hortonworks.com", "type": "all"
  }
},
{
  "metric": "tsd.http.latency_75pct", "timestamp": 1510603063, "value": "20", "tags": {
    "host": "sandbox.hortonworks.com", "type": "all"
  }
},
{
  "metric": "tsd.http.latency_90pct", "timestamp": 1510603063, "value": "70", "tags": {
    "host": "sandbox.hortonworks.com", "type": "all"
  }
}
```

Figura 3.10.: Salida de estadísticas.

```
response_content_type": "application/json; charset=UTF-8", "parsers": [
  "PutV1", "SuggestV1", "AnnotationV1", "AnnotationsV1",
  "AnnotationBulkDeleteV1", "TreeV1", "TreeRulesV1", "TreeTSUIDsListV1",
  "SearchQueryV1", "TreeRuleV1", "QueryV1", "LastPointQueryV1",
  "UidAssignV1", "UidMetaV1", "TSMetaV1", "UidRenameV1"], "serializer":
  "json", "class": "net.opentsdb.tsd.HttpJsonSerializer"]}
```

3.3.2.14. /api/stats.

Con este comando se devuelve una lista de estadísticas del funcionamiento del TSD. La petición puede ser de tipo GET ó POST. Por ejemplo, si en la barra de direcciones del navegador ponemos:

```
http://localhost:8091/api/stats
```

Una parte del resultado obtenido se puede ver en la figura 3.10.

3.3.2.15. /api/suggest.

Con esta instrucción se intenta emular un cierto sistema de «autocompletar» al que se puede acceder repetidamente a medida que el usuario ingresa una solicitud en una GUI. No ofrece búsquedas de texto completo o comodines, sino que simplemente coincida con las cadenas que se pasen en la consulta. Por ejemplo si en la línea de comandos de navegador se pone la expresión (lógicamente siguiendo al url correspondiente) «type=metrics&q=sys», devolverá las 25⁶ métricas que empiecen por «sys». Las peticiones con este comando pueden ser de tipo GET ó POST.

Los parámetros que se pueden usar son los siguientes:

- **type**. El tipo de dato para autocompletar, que puede tomar el valor de metrics, tagk ó tagv. **Requerido**: SI.
- **q** . Un String para delimitar los «types» requeridos. **Requerido**: NO. **Query string**: q .
- **max**. El número máximo de resultados devueltos. Debe ser un número mayor que cero. **Requerido**: No ; **Query string**: max.

⁶Valor por defecto y que se regula con el parámetro «max»

3. Trabajando con los datos de OpenTSDB.

En el ejemplo que hemos desarrollado en este trabajo, podemos poner en la ventana de direcciones de un navegador los siguiente:

```
http://localhost:8091/api/suggest?type=tagv&max=10
```

El resultado seria el siguiente:

```
["AAPL","GOOG","HDP","MSFT","ORCL"]
```

Ya que lo que se está demandando son los «tagv» que existen con una limitación de 10 elementos.

Si se hace una petición e tipo POST, el JSON utilizado sería:

```
{
    "type": "tagv",
    "max": 10
}
```

3.3.2.16. /api/tree.

No se desarrolla en este trabajo por quedar fuera del alcance del mismo.

3.3.2.17. /api/uid/assign.

Con esta instrucción se pueden asignar UIDs a las nuevas métricas, tag-clave y tag-valor. En una sola llamada, se pueden asignar múltiples valores. La petición puede ser de tipo GET ó POST.

Cada petición debe tener uno o más de los siguientes campos:

- **metric.** Una lista de los nombres de las métricas a asignar. **Query string:** metric.
- **tagk.** Una lista de tag conteniendo nuevas claves a ser utilizadas. **Query string:** tagk.
- **tagv.** Una lista contenido lo nuevos tags-valores . **Query string:** tagv.

Un ejemplo de este uso se puede ver a continuación:

```
http://localhost:8091/api/uid/assign?metric=sys2 , sys3 , sys4&tagk=t1 , t2
&tagv=web1 , web2
```

En formato JSON sería como a continuación de muestra:

```
{
    "metric": [
        "sys2",
        "sys3",
        "sys4"
    ],
    "tagk": [
        "t1"
    ],
    "tagv": [

```

3. Trabajando con los datos de OpenTSDB.

```
"web1",  
"web2"  
]  
}
```

La respuesta que se obtiene al ejecutar este código la podemos ver en el listado siguiente:

```
{"metric":{"sys2":"00000C","sys3":"00000D","sys4":"00000E"},"tagk":{"  
  t1":"000003","t2":"000004"},"tagv":{"web1":"000006","web2"  
  ":"000007"}}
```

3.3.2.18. /api/uid/tsmeta.

Con esta instrucción se puede añadir, borrar o modificar información de los metadatos de las series temporales. Recordemos que estos metadatos están asociados con una métrica y unos o varios tags. Esta instrucción se puede usar para peticiones GET, POST, PUT y DELETE.

Peticiones tipo GET.

Con este tipo de peticiones se consigue buscar entre los meta-objetos existentes, los que cumplen con los requisitos marcados. Existen dos tipos de peticiones de este tipo:

1. **tsuid**. Hay que facilitar un sólo valor hexadecimal igual al valor de metadato que se está buscando. El resultado incluye un solo objeto.
2. **métrica**. Aquí se facilitará el nombre de una métrica y uno o varios tags. El resultado va a ser un array de uno o más objetos.

Un ejemplo de la primera situación se muestra a continuación:

```
http://localhost:4242/api/uid/tsmeta?tsuid=00002A000001000001
```

Otro ejemplo para una petición de cierta métrica se da a continuación:

```
http://localhost:4242/api/uid/tsmeta?m=sys.cpu.nice&dc=lg
```

Peticiones de tipo POST/PUT.

Con este tipo de peticiones los campos que se pueden facilitar son los siguientes:

- **tsuid**. Es una representación hexadecimal de un UID de una serie temporal. **Tipo dato**: String; **Requerido**:SI; **Query string**: tsuid; **Ejemplo**: 00002A000001000001.
- **descripcion**. Una descripción breve de lo que representa el UID. **Tipo dato**: String; **Requerido**:NO; **Query string**: descripcion; **Ejemplo**: Un ejemplo de muestra.
- **displayName**. Un nombre corto que puede ser mostrado en GUIs en lugar del nombre por defecto. **Tipo dato**: String; **Requerido**:NO; **Query string**: display_name; **Ejemplo**: Nombre corto.

3. Trabajando con los datos de OpenTSDB.

- **notes.** Notas detalladas de lo que representa un determinado UID. **Tipo dato:** String; **Requerido:**NO; **Query string:** notes.
- **custom.** Un objeto de tipo map que contiene pares clave-valor para almacenar tags a medida. **Tipo dato:** Map; **Requerido:**NO; **Query string:** ⁷.
- **units.** Unidades utilizadas en los datos de las serie temporales almacenadas. **Tipo dato:** String; **Requerido:**NO; **Query string:** units; **Ejemplo:** Mbps.
- **dataType.** El tipo de dato almacenado en la serie temporal. Por ejemplo counter, gauge, absolute, etc. **Tipo dato:** String; **Requerido:**NO; **Query string:** data_type; **Ejemplo:** gauge.
- **max.** Un número opcional para el máximo de la serie temporal. Con un valor Nan se ignora. **Tipo dato:** Float; **Requerido:**NO; **V. defecto:**NaN; **Query string:** max.
- **min.** Igual que en el caso anterior pero para el mínimo. **Tipo dato:** Float; **Requerido:**NO; **V. defecto:**NaN; **Query string:** min.

Además de todos los campos anteriores, se pueden facilitar en un query string los siguientes parámetros:

- **m** .- Para facilitar un métrica.
- **create**.- Un flag con un valor igual a true.

A continuación se muestra un ejemplo:

```
http://localhost:4242/api/uid/tsmeta?display_name=Testing&m=sys.cpu.nice{
  host=web01,dc=lga}&create=true&method_override=post
```

Si ya existe el valor del TS meta objeto, entonces es sobrescrito con el enviado en este comando.

Un ejemplo de petición POST o PUT, se da a continuación:

```
http://localhost:4242/api/uid/tsmeta?tsuid=00002A000001000001&
  method_override=post&display_name=System%20CPU%20Time
```

El fichero JSON puede ser el siguiente:

```
{
  "tsuid": "00002A000001000001",
  "displayName": "System CPU Time for Webserver 01",
  "custom": {
    "owner": "Jane Doe",
    "department": "Operations",
    "assetTag": "12345"
  }
}
```

⁷Estos elementos a medida no se pueden pasar vía «query String», se deben usar para ello elementos de tipo POST o PUT

3. Trabajando con los datos de OpenTSDB.

Ejemplo de petición DELETE puede ser el siguiente:

```
http://localhost:4242/api/uid/tsmeta?tsuid=00002A000001000001&method_override=delete
```

Y un ejemplo de fichero JSON:

```
{  "tsuid":"00002A000001000001"}
```

3.3.2.19. /api/uid/uidmeta.

Con esta instrucción se puede editar o borrar información sobre los metadatos de los UID, que recordemos son los metadatos asociados con las métricas, tags-nombre y tags-valor. Este comando se puede utilizar con peticiones GET, POST, PUT y DELETE.

Los campos que se pueden utilizar con una petición con los siguientes:

- **uid**. La representación hexadecimal de un UID. **Tipo dato**: String; **Requerido**:SI; **Query string**: uid;**Ejemplo**: 2A.
- **type**. El tipo de elemento UID que puede ser: metric, tagk, tagv. **Tipo dato**: String; **Requerido**:SI; **Query string**: type;**Ejemplo**: metric.
- **descripcion**. Un descripción corta de lo que representa el UID. **Tipo dato**: String; **Requerido**:NO; **Query string**: description;**Ejemplo**: Esto es una descripción.
- **displayName**. Un nombre corto que puede mostrarse en los GUIs en lugar de su nombre por defecto. **Tipo dato**: String; **Requerido**:NO; **Query string**: display_name;**Ejemplo**: titulo corto.
- **notes**. Notas detalladas de lo que representa el UID. **Tipo dato**: String; **Requerido**:NO; **Query string**: notes ;**Ejemplo**: Describir los detalles.
- **custom**. Un objeto Map que contiene pares clave-valor a medida . **Tipo dato**: MAP; **Requerido**:NO.

El campo **custom** no se puede pasar vía query string, se deben usar los verbos POST o PUT para pasar sus valores.

Un ejemplo de petición GET se muestra a continuación:

```
http://localhost:4242/api/uid/uidmeta?uid=00002A&type=metric
```

Otro ejemplo de petición POST o PUT se muestra a continuación:

```
http://localhost:4242/api/uid/uidmeta?uid=00002A&type=metric&method=post&display_name=System%20CPU%20Time
```

El posible contenido del fichero JSON podría ser el siguiente:

```
{  "uid":"00002A",  "type":"metric",  "displayName":"System CPU Time",  "custom": {    "owner": "Jane Doe",
```

3. Trabajando con los datos de OpenTSDB.

```
[root@sandbox ~]# curl http://localhost:8091/api/version
{"short_revision":"95790c7","repo":"/root/opentsdb/build","host":"sandbox.hortoworks.com","version":"2.3.0","full_revision":"95790c73e09727905e2b971802c657c29c178efc","repo_status":"MINT","user":"root","branch":"master","timestamp":"1511511511"}
[root@sandbox ~]#
```

Figura 3.11.: Salida /api/version.

```
    "department ": "Operations",
    "assetTag ": "12345"
  }
}
```

Un ejemplo de petición DELETE también se puede ver aquí:

`http://localhost:4242/api/uid/uidmeta?uid=00002A&type=metric&method=delete`

Y el JSON:

```
{
  "uid ":"00002A",
  "type ":"metric"
}
```

3.3.2.20. /api/version

Con este comando se obtiene la versión de OpenTSDB con la que se está trabajando. Se puede hacer peticiones de tipo GET, POST. Un ejemplo de su uso en el shell de comandos es el siguiente:

```
curl http://localhost:8091/api/version
```

La salida obtenida de la ejecución del anterior comando se muestra en la figura 3.11.

4. Utilidades.

4.1. tcollector.

La utilidad tcollector es un proceso del lado del cliente que obtiene datos de índole diversa y los coloca cada cierto periodo de tiempo en la base de datos OpenTSDB.

4.1.1. Instalación de tcollector.

La instalación de esta herramienta es muy sencilla, y consiste básicamente en bajarse los colectores de GitHub. Para ello, se recomienda posicionarse en la carpeta «/root/opentsdb/» y ejecutar el siguiente comando:

```
git clone git://github.com/ub.com/OpenTSDB/tcollector.git
```

Simplemente con esta instrucción se creará una carpeta denominada «tcollector» que contendrá todos los elementos necesarios para utilizar esta herramienta. En la carpeta así creada, también existe otra subcarpeta denominada «collectors» que a su vez contiene otra serie de subcarpetas con una denominación numérica. Así por ejemplo si hay una carpeta denominada «15», los programas dentro de la misma (que serán los colectores de datos), se ejecutarán periódicamente cada 15 minutos.

Hay también una carpeta con denominación «0» de tal manera que los colectores en ellas depositados se estarán ejecutando continuamente.

Una vez instalado todo este sistema de colectores como se ha indicado anteriormente, para su puesta en marcha, hay que situarse en la carpeta raíz de la instalación y ejecutar la siguiente instrucción : «./tcollector <start [args] | stop |restart [args] | status>». Después ejecutar el programa python con la siguiente instrucción: «sudo python tcollector.py -H <TSDB Host IP> -p<TSDB port> -D». Con el parámetro -H se indica el host donde está instalado OpenTSDB, con -p se indicará el puerto, y por último -D es para indicar que se ejecute como un demonio en background¹.

Una vez ejecutado todo este sistema, por defecto se crea una salida log en el fichero localizado en «/var/log/tcollector.log». Esta ubicación y fichero se puede cambiar editando el fichero denominado «tcollector.py». Estos parámetros se muestran en la figura 4.1.

La propia instalación viene por defecto con una serie de colectores listos para su uso de forma inmediata. Las lista de colectores y la indicación de lo que genera cada uno de ellos, se puede encontrar en el siguiente enlace: http://opentsdb.net/docs/build/html/user_guide/utilities/tcollector.html

¹Estas instrucciones son válidas para la versión 2.3 que es con la que se ha estado trabajando para el desarrollo del presente trabajo. Para versiones anteriores, y de acuerdo con las instrucciones existentes en la web de OpenTSDB, habría que editar el fichero «tcollector/startstop» y poner el valor corrector en la variable de configuración: TSD_HOST=dns.name.of.tsd.

4. Utilidades.

```
# global variables.
COLLECTORS = {}
GENERATION = 0
DEFAULT_LOG = '/var/log/tcollector.log'
LOG = logging.getLogger('tcollector')
ALIVE = True
# If the SenderThread catches more than this many consecutive uncaught
# exceptions, something is not right and tcollector will shutdown.
# Hopefully some kind of supervising daemon will then restart it.
MAX_UNCAUGHT_EXCEPTIONS = 100
DEFAULT_PORT = 4242
MAX_REASONABLE_TIMESTAMP = 1600000000 # Good until September 2020 :)
# How long to wait for datapoints before assuming
# a collector is dead and restarting it
ALLOWED_INACTIVITY_TIME = 600 # seconds
MAX_SENDQ_SIZE = 10000
MAX_READQ_SIZE = 100000

def register_collector(collector):
    """Register a collector with the COLLECTORS global"""
```

Figura 4.1.: Parámetros por defecto de tcollector.

```
(D:\programas\Anaconda\envs\py27) C:\Users\Francisco>pip install potsdb
Collecting potsdb
  Downloading potsdb-1.0.3.tar.gz
Building wheels for collected packages: potsdb
  Running setup.py bdist_wheel for potsdb ... done
  Stored in directory: C:\Users\Francisco\AppData\Local\pip\Cache\wheels\c4\eb\81\e8d8da8c1ce64bd2e9c66194d89ff9dc4184a2c48c2ecba811
Successfully built potsdb
Installing collected packages: potsdb
Successfully installed potsdb-1.0.3

(D:\programas\Anaconda\envs\py27) C:\Users\Francisco>
```

Figura 4.2.: Instalación de Potsdb

4.2. Potsdb.

Este constituye un paquete de python que se puede encontrar [haciendo clic en este enlace](#). La instalación se efectúa de una forma sencilla, tan solo hay que teclear en la línea de comando lo siguiente: «pip install potsdb» tal y como se muestra en la figura 4.2².

Este paquete se puede utilizar para enviar datos al servidor OpenTSDB. Un ejemplo de esta situación se puede ver en el siguiente ejemplo:

```
import potsdb
metrics=potsdb.Client('localhost',port=8091,mps=100)
#Otros valores que se pueden incluir como parámetros son:
#qsize: Tamaño máximo de la cola
#host_tag: True para automático; una cadena caracteres para
sobreescritura, None para nada
```

²Para el desarrollo de este apartado, la instalación de ha hecho sobre python 2.7

4. Utilidades.

```
#mps: Métricas por segundo.
#check_host: False si no se quiere chequear la conectividad.

#Para mandar un valor con tiempo el actual, metrica la indicada en
  primer parámetro
#valor el indicado en el segundo parámetro.
metrics.send('close',200)
#Se pueden indicar también tags..
metrics.send('close',300,type='GOOG')
```

4.3. Cliente R.

Para el paquete estadístico R también se puede encontrar un paquete que permite la lectura de los datos de OpenTSDB. En el siguiente listado, se ofrece un ejemplo de código para poder conectar con OpenTSDB con los datos cargados según lo comentado en el apartado 1.3 en la página 16 y extraer la serie temporal que se desee.

```
if (!require("stringr")) install.packages("stringr");require("stringr")
if (!require("httr")) install.packages("httr");require("httr")
if (!require("rjson")) install.packages("rjson");require("rjson")
if (!require("lubridate")) install.packages("lubridate");require("lubridate")
if (!require("data.table")) install.packages("data.table");require("data.table")
if (!require("devtools")) install.packages("devtools")
library("devtools")
install_github("opentsdbr", "holstius")
library(opentsdbr)

start<-interval(ymd_hms("2017-11-28 16:00:00"),ymd_hms("2017-11-29 20:00:00"))
(result <- tsd_req(metric='open', interval=start, tags=c(symbol="GOOG"),hostname='localhost',port = 8091))
```

Obteniendo el siguiente resultado:

	metric	timestamp	value	symbol
1:	open	2017-11-28 16:00:00	1054.106	GOOG
2:	open	2017-11-28 16:01:00	1053.340	GOOG
3:	open	2017-11-28 16:02:00	1053.480	GOOG
4:	open	2017-11-28 16:03:00	1054.000	GOOG
5:	open	2017-11-28 16:04:00	1053.330	GOOG
<hr/>				
616:	open	2017-11-29 19:56:00	1020.490	GOOG
617:	open	2017-11-29 19:57:00	1020.275	GOOG
618:	open	2017-11-29 19:58:00	1022.120	GOOG
619:	open	2017-11-29 19:59:00	1021.849	GOOG
620:	open	2017-11-29 20:00:00	1021.440	GOOG

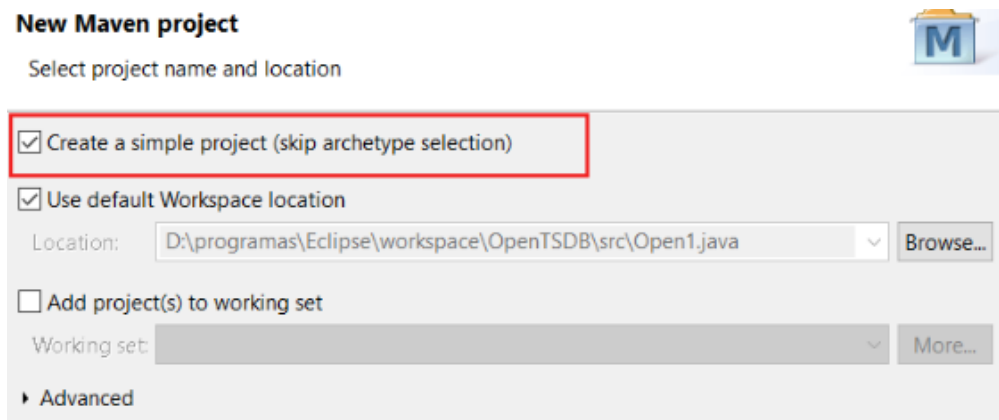


Figura 4.3.: Selección sin arquetipo.

4.4. Clientes con Java.

Java es un lenguaje de programación ampliamente Othe entre el mundo de desarrolladores. Además Java presenta la ventaja de poseer clases que permiten crear sockets y enlaces HTML de una forma fácil y eficiente. Por este motivo, este lenguaje de programación se presta muy bien al desarrollo de programas que enlazan con la base de datos OpenTSDB. En este apartado vamos a presentar dos tipos de desarrollo utilizando esta herramienta de programación y apoyándonos en el IDE Eclipse.

4.4.1. Opentsdb-java-client.

Es un proyecto de cliente Java sobre OpenTSDB [que se encuentra el github](#). Con este cliente se podrán enviar métricas y realizar queries sobre OpenTSDB, de una forma fácil. Para el desarrollo de esta exposición se ha utilizado la herramienta gratuita de desarrollo eclipse, por entender que facilita mucho la planificación de desarrollo.

En primer lugar lo que hay que hacer es bajarse de github el proyecto en formato de fichero zip. Posteriormente se descomprime el fichero bajado y se creará de esta manera la estructura necesaria de directorios que se necesita. Posteriormente, y dentro de eclipse se creará un proyecto «maven», que denominaremos «prueba_opentsdb». Por lo tanto se seleccionará : File > New > Other > Maven Project y como arquetipo, se seleccionará la opción que figura en la figura 4.3, de tal manera que el arquetipo inicial que se cree se simple, ya que posteriormente el fichero POM será modificado por el que está en el proyecto github que se ha bajado.

En la figura 4.4, se muestra la pantalla donde hay que poner la identificación del proyecto maven que se está creando.

A continuación lo que procede es importar en nuestro proyecto maven, el proyecto github que se ha descargado, descomprimido y guardado en algún lugar del disco duro del sistema.

Para conseguir esta importación, se hace clic con el botón derecho del ratón sobre

4. Utilidades.

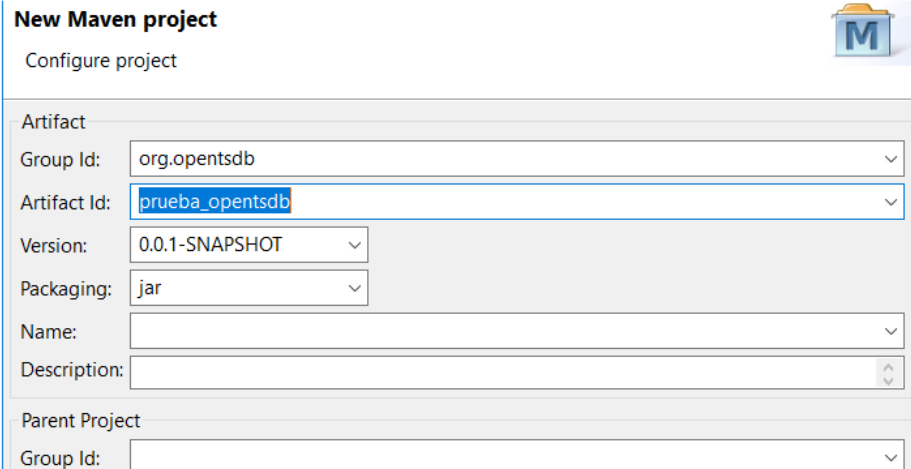


Figura 4.4.: Denominación del proyecto maven.

el nombre del proyecto y se selecciona importar en el menú emergente que aparece. A continuación, se abre la opción «General» y después «File System» dando a continuación en el botón «next».

Esto nos llevará a una pantalla como la que se muestra en la figura 4.5. Después de haber hecho clic en el botón «Browse» se ha podido seleccionar la carpeta donde se ha descargado el proyecto de github y se ha seleccionado la casilla del proyecto. También se ha marcado la opción «Overwrite existing resource without warning». A continuación se hace clic en el botón «Finish» y ya se tendría importado todos los paquetes en nuestro proyecto.

Una vez importado estos ficheros, se puede ver que en el IDE de eclipse aparecen marcas de posibles errores. Ello es debido a que no se ha actualizado el proyecto «prueba_opentsdb», para ello se hace clic derecho sobre el nombre del proyecto y se hace la siguiente selección: Maven > Update Project ... Una vez hecho esto se puede ver que los errores ha desaparecido.

A continuación se creará una clase en la carpeta «src/main/java» que tenga el método «main» y con el siguiente contenido:

```
import java.io.IOException;

import org.opentsdb.client.ExpectResponse;
import org.opentsdb.client.HttpClient;
import org.opentsdb.client.HttpClientImpl;
import org.opentsdb.client.builder.MetricBuilder;
import org.opentsdb.client.response.Response;
```

4. Utilidades.

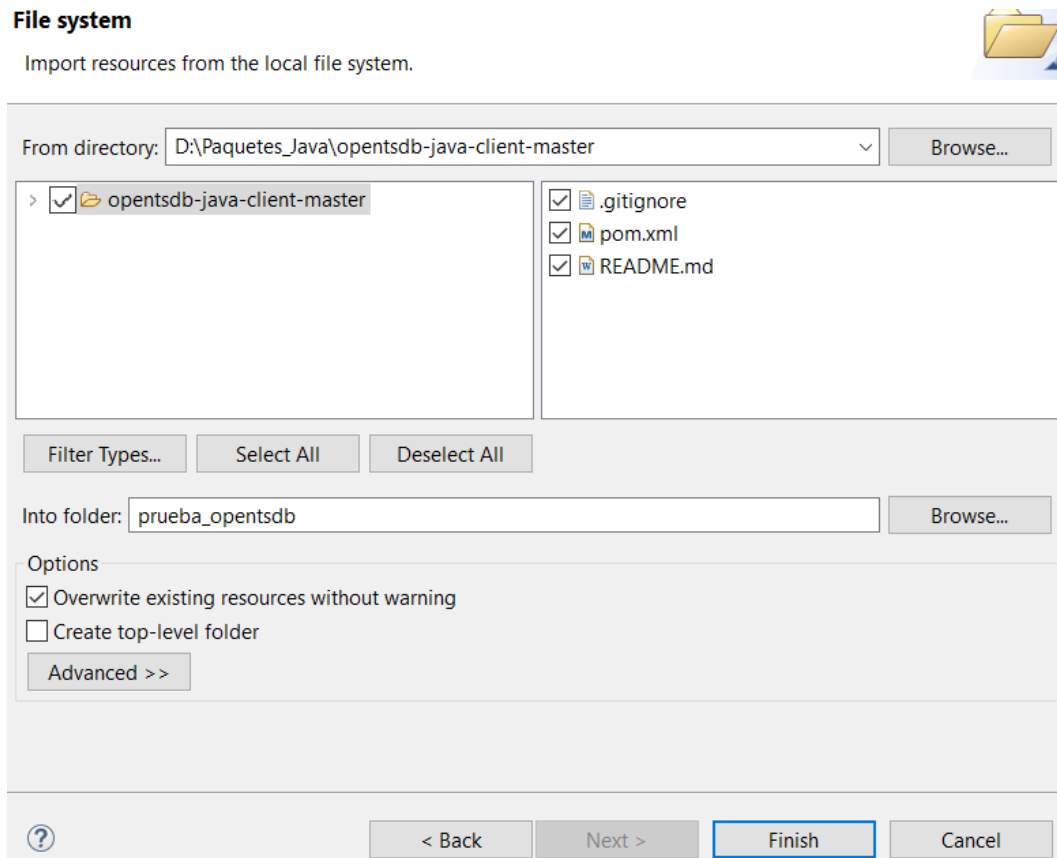


Figura 4.5.: Importar proyecto cliente OpenTSDB.

4. Utilidades.

```
public class principal {  
  
    public static void main(String[] args) {  
        HttpClient client = new HttpClientImpl("http://  
            localhost:8091");  
  
        MetricBuilder builder = MetricBuilder.getInstance();  
  
        builder.addMetric("open").setDataPoint(2,30L).addTag  
            ("type", "GOOG");  
  
        try {  
            Response response = client.pushMetrics(  
                builder,  
                    ExpectResponse.DETAIL);  
            System.out.println(response);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

De esta manera ya se tendría construido un pequeño proyecto con código Java para poder enviar datos a OpenTSDB.

Como nota final de este apartado cabe decir lo siguiente. Hay que tener la precaución de que el proyecto lo compile de forma automática eclipse, para ello hay que seleccionar el proyecto y después en el sistema de menús de eclipse, seleccionar el apartado «Project» y ver que está marcada la casilla «Build Automatically». Para verificar que las clases correspondientes se han generado, se puede acudir al navegador de windows al workspace de eclipse y ahí seleccionar la carpeta del proyecto. En la carpeta «target» deberán estar todas las clases generadas.

4.4.2. Construir un cliente vía la clase HttpURLConnection.

Otra forma de construir clientes OpenTSDB con Java es utilizando la clase «HttpURLConnection». Un ejemplo de su uso se puede ver en el listado del código que a continuación se muestra.

```
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.OutputStreamWriter;  
import java.io.UnsupportedEncodingException;  
import java.net.HttpURLConnection;  
import java.net.URL;  
import org.json.*;  
  
public class Open1 {
```

4. Utilidades.

```
public static void main(String[] args) throws IOException {
    URL url=new URL("http://localhost:8091/api/query/");
    HttpURLConnection conn=(HttpURLConnection) url.openConnection();
    conn.setRequestProperty("Accept", "application/json");
    conn.setRequestProperty("Content-type", "application/json");
    conn.setRequestMethod("POST");
    conn.setDoOutput(true);
    conn.setDoInput(true);
    OutputStreamWriter writer=new OutputStreamWriter(conn.getOutputStream());
    long startEpoch=1420088400L;
    long endEpoch=1425501345L;
    String metric="close";
    String aggregator="sum";
    JSONObject mainObject=new JSONObject();
    mainObject.put("start", startEpoch);
    mainObject.put("end", endEpoch);
    JSONArray queryArray=new JSONArray();
    JSONObject queryParams=new JSONObject();
    queryParams.put("aggregator", aggregator);
    queryParams.put("metric", metric);
    queryArray.put(queryParams);
    //Para los tags
    JSONObject queryTags=new JSONObject();
    queryTags.put("type", "GOOG");
    queryParams.put("tags", queryTags);
    mainObject.put("queries", queryArray);
    String queryString=mainObject.toString();
    System.out.println(queryString);
    writer.write(queryString);
    writer.flush();
    writer.close();
    //Para chequear el código de respuesta.
    int HttpResult=conn.getResponseCode();
    if (HttpResult==HttpURLConnection.HTTP_OK ) {
        System.out.println("datos Guardados");
    }
}

public static String readHTTPConnection(HttpURLConnection conn2) {
    StringBuilder sb=new StringBuilder();
    BufferedReader br;
    try {
        br=new BufferedReader(new InputStreamReader(conn2.getInputStream()), "
");
        String line=null;
        while((line=br.readLine())!= null) {
            sb.append(line+"\n");
        }
        br.close();
    }catch(UnsupportedEncodingException e) {
    }
```


4. Utilidades.

```
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return sb.toString();
}
}
```

https://bigdatafran.github.io/big_data/

A. Código fichero google_intraday.py.

A continuación se muestra el código del fichero python google_intraday.py.

```
# Code taken from http://trading.cheno.net/wp-content/uploads/2011/12/google\_intraday.py
# See for more info: http://trading.cheno.net/downloading-google-intraday-historical-data-with-python/
# Copyright (c) 2011, Mark Chenoweth
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted
# provided that the following conditions are met:
#
# - Redistributions of source code must retain the above copyright
#   notice, this list of conditions and the following disclaimer.
#
# - Redistributions in binary form must reproduce the above copyright
#   notice, this list of conditions and the following
#   disclaimer in the documentation and/or other materials provided
#   with the distribution.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
# "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES,
# INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
# BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
# EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
# PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
# OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED
# AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
# STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
# ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

from __future__ import print_function
import urllib, time, datetime, sys
```

```
class Quote(object):
```

```
    DATE_FMT = '%Y-%m-%d'
```

A. Código fichero google_intraday.py.

```

TIME_FMT = '%H: %M: %S'

def __init__(self):
    self.symbol = ''
    self.date, self.time, self.open_, self.high, self.low, self.close, self
        .volume = ([] for _ in range(7))

def append(self, dt, open_, high, low, close, volume):
    self.date.append(dt.date())
    self.time.append(dt.time())
    self.open_.append(float(open_))
    self.high.append(float(high))
    self.low.append(float(low))
    self.close.append(float(close))
    self.volume.append(int(volume))

def to_csv(self):
    return ''.join(["{0},{1},{2},{3:.2f},{4:.2f},{5:.2f},{6:.2f},{7}\n"
        .format(self.symbol,
            self.date[bar].strftime('%Y-%m-%d'), self.time[bar].
                strftime('%H: %M: %S'),
            self.open_[bar], self.high[bar], self.low[bar], self.close
                [bar], self.volume[bar])
            for bar in xrange(len(self.close))])

def write_csv(self, filename):
    with open(filename, 'w') as f:
        f.write(self.to_csv())

def read_csv(self, filename):
    self.symbol = ''
    self.date, self.time, self.open_, self.high, self.low, self.close, self
        .volume = ([] for _ in range(7))
    for line in open(filename, 'r'):
        symbol, ds, ts, open_, high, low, close, volume = line.rstrip().split(
            ',')
        self.symbol = symbol
        dt = datetime.datetime.strptime(ds+'_'+ts, self.DATE_FMT+'_'+
            self.TIME_FMT)
        self.append(dt, open_, high, low, close, volume)
    return True

def __repr__(self):
    return self.to_csv()

class GoogleIntradayQuote(Quote):
    ''' Intraday quotes from Google. Specify interval seconds and
        number of days '''

```

A. Código fichero google_intraday.py.

```
def __init__(self, symbol, interval_seconds=300, num_days=5):
    super(GoogleIntradayQuote, self).__init__()
    self.symbol = symbol.upper()
    url_string = "https://finance.google.com/finance/getprices?q={0}"
        .format(self.symbol)
    url_string += "&i={0}&p={1}d&f=d,o,h,l,c,v".format(
        interval_seconds, num_days)
#    print(url_string)
    csv = urllib.urlopen(url_string).readlines()
    for bar in xrange(7, len(csv)):
        if csv[bar].count(',') != 5: continue
        offset, close, high, low, open_, volume = csv[bar].split(',')
        if offset[0] == 'a':
            day = float(offset[1:])
            offset = 0
        else:
            offset = float(offset)
        open_, high, low, close = [float(x) for x in [open_, high, low, close]]
        dt = datetime.datetime.fromtimestamp(day + (interval_seconds *
            offset))
        self.append(dt, open_, high, low, close, volume)

if __name__ == '__main__':

    f = open('opentsd.input', 'a')

    if len(sys.argv) > 1:
        stocks = sys.argv[1]
    else:
        stocks = ("AAPL")
    for stock in stocks.split(','):
        if 'quotes' in locals():
            del(quotes)
        quotes = GoogleIntradayQuote(stock, 60, 300)
        #atts = [a for a in dir(quotes) if not a.startswith('__')]
        #print atts
        for i in range(len(quotes.high)):
            #from dateutil import parser
            #datetime_str = str(quotes.date[i]) + ' ' + str(quotes.time[i])
            # + ' PST'
            #dt = parser.parse(datetime_str)
            from datetime import datetime
            datetime_str = str(quotes.date[i]) + '-' + str(quotes.time[i])

            dt = datetime.strptime(datetime_str, "%Y-%m-%d-%H:%M:%S")
            td = (dt - datetime(1970, 1, 1))
```

A. Código fichero google_intraday.py.

```
epoch_time = (td.microseconds + (td.seconds + td.days * 86400)
              * 10**6) / 10**6

#print 'open ' + str(epoch_time) + ' ' + str(quotes.date[i])
#      + ' ' + str(quotes.time[i]) + ' ' + str(quotes.open_[i]) +
#      ' symbol=' + str(quotes.symbol)
print ('open_' + str(epoch_time) + '_' + str(quotes.open_[i])
      + '_symbol=' + str(quotes.symbol), file=f)
print ('close_' + str(epoch_time) + '_' + str(quotes.close[i])
      + '_symbol=' + str(quotes.symbol), file=f)
print ('high_' + str(epoch_time) + '_' + str(quotes.high[i]) +
      '_symbol=' + str(quotes.symbol), file=f)
print ('low_' + str(epoch_time) + '_' + str(quotes.low[i]) + '
      _symbol=' + str(quotes.symbol), file=f)
print ('volume_' + str(epoch_time) + '_' + str(quotes.volume[i]
      ]) + '_symbol=' + str(quotes.symbol), file=f)

print (quotes)
f.close()
```

B. Google Finance.

A pesar de que este apartado no queda encuadrado por su contenido dentro del tema principal del presente trabajo, dado que se ha utilizado las prestaciones de Google finance en el anterior apartado, en lo que sigue se procede a exponer de forma resumida el contenido de esta herramienta gratuita proporcionada por Google.

Para entrar en este servicio, se deberá utilizar la siguiente dirección web <https://finance.google.com/>, donde existe un buscador para poder localizar cotización de valores de bolsas mundiales. No obstante lo atractivo de este servicio es que permite bajar cotización de valores bursátiles (junto a otros datos como apertura, cierre, volumen, etc) mediante programación, lo cual permite tratar estos valores con herramientas por ejemplo estadísticas y poder obtener conclusiones de indudable valor para la inversión bursátil.

Para obtener esos valores, se deberá utilizar una URL similar a la que a modo de ejemplo se muestra a continuación:

```
http://www.google.com/finance/getprices?q=GOOG&x=NASD&i=86400&p=40Y&f=d,c,v,k,o,h,l&df=cpc&auto=0&ei=Ef6XUYDfCqSTiAKEMg
```

En el anterior ejemplo de enlace, se puede ver que después de la expresión¹ «http://www.google.com/finance» aparecen una serie de pares de tipo clave-valor separados por el símbolo «&» cuyo significado es el siguiente:

- q - Para indicar el símbolo del valor bursátil que se quiere bajar.
- x - Mercado en el que cotiza el valor bursátil (ex: NASD).
- i - Tamaño del intervalo en segundos (86400 = 1 día de intervalo)
- p - Periodo. (Un número seguido por una "d" o "Y", ej. 40Y = 40 años.)
- f - Para indicar lo que se quiere bajar d (date - timestamp/interval, c - cierre, v - volumen, etc...)
- df - ??
- auto - ??
- ei - ??
- ts - Comienzo del tiempo (Unix format). Si blanco, comienza por el día en el que se pide.

¹Observar que la tipología de este enlace es una expresión URL de tipo GET para enviar información añadida al servidor.

Índice alfabético

A

Annotations, 26
assign, 32

B

BigData, 5

C

CLI, 29

D

data poin, 6
delete, 32
diediedie, 38
downsample, 49
Downsampling, 22, 49
dropcaches, 38

E

epoch, 27
Explicit Tags., 40

F

Filtros, 24, 49
fsck, 33, 36
funciones de agregación, 22

G

gnuplot, 17
Google finance, 70
Grafana, 17
grep, 31

H

HBASE, 5
HDFS, 5
help, 37

histogram, 37

I

import, 34

M

Metadatos, 26
metapurge., 33
metasync, 33
mkmetric, 34

O

openTSDB, 5
opentsdb.conf, 19

P

Potsdb, 58
Pre-aggregates, 24
put, 37
Putty, 9

Q

query, 35

R

Rate, 23
Rate Options, 49
rename, 32
RESTful, 38
Rollup, 24
rollup, 37

S

scan, 36
search, 36
stats, 37
Sub Queries, 48

Sub QueriesSub Queries, 48

T

tagk, 25

tagv, 25

tcollector, 57

treepurge, 34

treesync., 34

TSD, 5

tsd, 36

TSMeta, 26

TSUID, 25, 26

U

UID, 25

uid, 31

UIDMeta, 26

V

version, 37

W

wildcard, 39

Z

ZooKeeper, 12

zookeeper.znode.parent, 14

zookeeper.znode.parentzookeeper.znode.parent,
14