NAME / NOMBRE

rrdtutorial − Tutorial sobre RRDtool por Alex van den Bogaerdt (Traducido al castellano por Jesús Couto Fandiño)

DESCRIPTION / DESCRIPCIÓN

RRDtool es un programa escrito por Tobias Oetiker con la colaboración de muchas personas en diversas partes del mundo. Alex van den Bogaerdt escribió este documento para ayudarte a entender que es RRDtool y que es lo que puede hacer por ti.

La documentación que viene con RRDtool puede ser demasiado técnica para algunos. Este tutorial existe para ayudarte a entender las funciones básicas de RRdtool. Debe servirte de preparación para leer la documentación, y además explica algunas ideas generales sobre estadística, con un enfoque particular hacia las redes.

TUTORIAL

Importante

¡Por favor, no te adelantes en la lectura de este documento! Esta primera parte explica los fundamentos básicos. Puede ser aburrida, pero si te saltas los fundamentos, los ejemplos no te van a tener mucho sentido.

¿Qué es RRDtool?

RRDtool significa ‘‘herramienta de bases de datos en round robin’’. ‘‘Round robin’’ es una técnica que implica un número fijo de datos, y un apuntador al elemento más reciente. Piensa en un circulo con unos cuantos puntos dibujados alrededor del borde; estos puntos son los lugares donde se pueden guardar los datos. Dibuja ahora una flecha desde el centro del círculo a uno de los puntos; este es el apuntador. Cuando se lee o escribe el dato actualmente apuntado, la flecha se mueve al próximo elemento. Como estamos en un círculo, no hay ni principio ni fin; siempre puedes seguir, eternamente. Al cabo de un tiempo ya se habrán usado todas las posiciones disponibles y el proceso empieza a reutilizar las antiguas. De esta forma, la base de datos no crece en tamaño y, por lo tanto, no requiere ningún mantenimiento. RRDtool trabaja con estas bases de datos en ‘‘round-robin’’, guardando y recuperando datos de ellas.

¿Qué datos pueden guardarse en una RRD?

Lo que se te ocurra. Debes poder medir algún valor dado en distintos momentos en el tiempo y proveer a RRDtool de estos valores. Si puedes hacer esto, RRDtool puede guardar los datos. Los valores tienen que ser numéricos, pero no necesariamente enteros, como en MRTG.

Muchos ejemplos mencionan SNMP, que es el acrónimo de ‘‘Simple Network Management Protocol’’ (Protocolo Simple de Administración de Redes). Lo de ‘‘simple’’ se refiere al protocolo − no se supone que sea fácil administrar o monitorizar una red. Cuando hayas terminado con este documento, deberás saber lo suficiente para entender cuando oigas a otros hablar sobre SNMP. Por ahora, simplemente considera a SNMP como una forma de preguntarle a los dispositivos por los valores de ciertos contadores que mantienen. Son estos valores de estos contadores los que vamos a almacenar en la RRD.

¿Qué puedo hacer con esta herramienta?

RRDtool se deriva de MRTG (Multi Router Traffic Grapher, Graficador De Tráfico de Múltiples Enrutadores). MRTG empezó como un pequeño script para poder graficar el uso de una conexión a la Internet. Luego evolucionó, permitiendo graficar otras fuentes de datos, como temperatura, velocidad, voltajes, cantidad de páginas impresas, etc... Lo más probable es que empieces a usar RRDtool para guardar y procesar datos conseguidos a través de SNMP, y que los datos sean el número de bytes (o bits) transferidos desde y hacia una red u ordenador. RRDtool te permite crear una base de datos, guardar los datos en ellas, recuperarlos y crear gráficos en formato GIF o PNG, para mostrarlos en un navegador web. Esas imágenes dependen de los datos que hayas guardado y pueden, por ejemplo, ser un sumario del promedio de uso de la red, o los picos de tráfico que ocurrieron. También lo puedes usar para mostrar el nivel de las mareas, la radiación solar, el consumo de electricidad, el número de visitantes en una exposición en un momento dado, los niveles de ruido cerca del aeropuerto, la temperatura en tu lugar de vacaciones favorito, o en la nevera, o cualquier otra cosa que te puedas imaginar, mientras tengas algún sensor con el cual medir los datos y seas capaz de pasarle los números a RRDtool.

¿Y si aún tengo problemas después de leer este documento?

Lo primero, ¡léelo otra vez!. Puede que te hayas perdido de algo. Si no puedes compilar el código fuente y usas un sistema operativo bastante común, casi seguro que no es la culpa de RRDtool. Probablemente consigas versiones pre-compiladas por la Internet. Si provienen de una fuente confiable, úsalas. Si, por otro lado, el programa funciona, pero no te da los resultados que tu esperabas, puede ser un problema con la configuración; revísala y compárala con los ejemplos.

Hay una lista de correo electrónico y una archivo de la misma. Lee la lista durante unas cuantas semanas, y busca en el archivo. Es descortés hacer una pregunta sin haber revisado el archivo; ¡puede que tu problema ya haya sido resuelto antes! Normalmente ocurre así en todas las listas de correo, no sólo esta. Examina la documentación que vino con RRDtool para ver donde está el archivo y como usarlo.

Te sugiero que te tomes un momento y te subscribas a la lista ahora mismo, enviando un mensaje a rrd-users-request@list.ee.ethz.ch con título `subscribe`. Si eventualmente deseas salirte de la lista, envía otro correo a la misma dirección, con título `unsubscribe`.

¿Cómo me vas a ayudar?

Dándote descripciones y ejemplos detallados. Asumimos que el seguir las instrucciones en el orden en que se presentan aquí te dará suficiente conocimiento de RRDtool como para que experimentes por tu cuenta. Si no funciona a la primera, puede que te hallas saltado algo; siguiendo los ejemplos obtendrás algo de experiencia práctica y, lo que es más importante, un poco de información sobre como funciona el programa.

Necesitarás saber algo sobre números hexadecimales. Si no, empieza por leer ''bin_dec_hex'' antes de continuar.

Tu primera base de datos en round-robin

En mi opinión, la mejor forma de aprender algo es haciéndolo. ¿Por qué no empezamos ya? Vamos a crear una base de datos, poner unos cuantos valores en ella y extraerlos después. La salida que obtengas debe ser igual a la que aparece en este documento.

Empezaremos con algo fácil, comparando un coche con un enrutador, o por decirlo de otra forma, comparando kilómetros con bits y bytes. A nosotros nos da lo mismo; son unos números obtenidos en un espacio de tiempo.

Asumamos que tenemos un dispositivo que transfiere bytes desde y hacia la Internet. Este dispositivo tiene un contador que empieza en 0 al encenderse y se incrementa con cada byte transferido. Este contador tiene un valor máximo; si ese valor se alcanza y se cuenta un byte más, el contador vuelve a empezar desde cero. Esto es exactamente lo mismo que pasa con muchos contadores, como el cuentakilómetros del coche. En muchas de las disertaciones sobre redes se habla de bits por segundo, así que empezaremos por acostumbrarnos a esto. Asumamos que un byte son 8 bits y empecemos a pensar en bits y no en bytes. ¡El contador, sin embargo, sigue contando en bytes! En el mundo SNMP, la mayoría de los contadores tienen una longitud de 32 bits. Esto significa que pueden contar desde 0 hasta 4294967295. Usaremos estos valores en los ejemplos. El dispositivo, cuando le preguntamos, retorna el valor actual del contador. Como sabemos el tiempo transcurrido desde la última vez que le preguntamos, sabemos cuantos bytes se han transferido `***en promedio***` por segundo. Esto no es muy difícil de calcular; primero en palabras, luego en operaciones:

1. Toma el valor actual del contador y réstale el valor anterior

2. Haz lo mismo con la fecha

lo multiplicas por ocho obtienes la cantidad de bits por segundo

3. Divide el resultado del paso (1) por el resultado del paso (2). El resultado es la cantidad de bytes por segundo. Si

```
bps = (contador_actual - contador_anterior) / (fecha_actual - fecha_anterior) * 8
```

Para algunos será de ayuda traducir esto a un ejemplo automotor. No prueben estas velocidades en la práctica, y si lo hacen, no me echen la culpa por los resultados.

Usaremos las siguientes abreviaturas:

```
M:     metros
KM:    kilómetros (= 1000 metros).
H:     horas
S:     segundos
KM/H: kilómetros por hora
M/S:  metros por segundo
```

Vas conduciendo un coche. A las 12:05, miras el contador en el salpicadero y ves que el coche ha recorrido 12345 KM. A las 12:10 vuelves a mirar otra vez, y dice 12357 KM. Quiere decir, que has recorrido 12 KM en cinco minutos. Un científico convertiría esto en metros por segundos; esto es bastante parecido al problema de pasar de bytes transferidos en 5 minutos a bits por segundo.

Viajamos 12 kilómetros, que son 12000 metros. Tardamos 5 minutos, o sea 300 segundos. Nuestra velocidad es 12000M / 300S igual a 40 M/S.

También podemos calcular la velocidad en KM/H: 12 veces 5 minutos es una hora, así que multiplicando los 12 KM por 12 obtenemos 144 KM/H. No intentes esto en casa, o por donde vivo :–)

Recuerda que estos números son tan sólo promedios. No hay forma de deducir, viendo sólo los números, si fuiste a una velocidad constante. Hay un ejemplo más adelante en el tutorial que explica esto.

Espero que entiendas que no hay diferencia entre calcular la velocidad en M/S o bps; sólo la forma en que recogemos los datos es distinta. Inclusive, la K de kilo en este caso es exactamente la misma, ya que en redes k es 1000

Ahora vamos a crear una base de datos en la que guardar todos estos interesantes valores. El método a usar para arrancar el programa puede variar de un sistema de operación a otro, pero asumamos que lo puedes resolver tu mismo en caso que se diferente en el sistema que usas. Asegúrate de no sobreescribir ningún archivo en tu sistema al ejecutarlo y escribe todo como una sola línea (tuve que partirlo para que fuera legible), saltándote todos los caracteres '\'

```
rrdtool create test.rrd             \
         --start 920804400          \
         DS:speed:COUNTER:600:U:U   \
         RRA:AVERAGE:0.5:1:24       \
         RRA:AVERAGE:0.5:6:10
```

(o sea, escribe: `rrdtool create test.rrd --start 920804400 DS ...`)

¿Qué hemos creado?

Hemos creado una base de datos en round robin llamada test (test.rrd), que empieza desde el mediodía del día en que empecé a escribir este documento (7 de marzo de 1999). En ella se guarda una fuente de datos (DS), llamada ''speed'', que se lee de un contador. En la misma base de datos se guardan dos archivos en round robin (RRAs), uno promedia los datos cada vez que se leen (o sea, no hay nada que promediar), y mantiene 24 muestras (24 por 5 minutos = 2 horas de muestras). El otro promedia 6 muestras (media hora), y guarda 10 de estos promedios (o sea, 5 horas). Las opciones restantes las veremos más adelante.

RRDtool usa un formato de ''fecha'' especial que viene del mundo de UNIX. Estas ''fechas'' son el número de segundos que han pasado desde el primero de enero de 1970, zona UTC. Este número de segundos se convierte luego en la fecha local, por lo que varia según la franja horaria.

Lo más probable es que tu no vivas en la misma parte del mundo que yo, por lo que tu franja horaria será diferente. En los ejemplos, cuando mencione horas, puede que no sean las mismas para ti; esto no afecta mucho los resultados, sólo tienes que corregir las horas mientras lees. Por ejemplo, las 12:05 para mí son las 11:05 para los amigos en la Gran Bretaña.

Ahora tenemos que llenar nuestra base de datos con valores. Vamos a suponer que leímos estos datos:

```
12:05  12345 KM
12:10  12357 KM
12:15  12363 KM
12:20  12363 KM
12:25  12363 KM
12:30  12373 KM
12:35  12383 KM
12:40  12393 KM
12:45  12399 KM
12:50  12405 KM
12:55  12411 KM
13:00  12415 KM
13:05  12420 KM
13:10  12422 KM
13:15  12423 KM
```

Llenaremos la base de datos así:

```
rrdtool update test.rrd 920804700:12345 920805000:12357 920805300:12363
rrdtool update test.rrd 920805600:12363 920805900:12363 920806200:12373
rrdtool update test.rrd 920806500:12383 920806800:12393 920807100:12399
rrdtool update test.rrd 920807400:12405 920807700:12411 920808000:12415
rrdtool update test.rrd 920808300:12420 920808600:12422 920808900:12423
```

Lo que significa: actualiza nuestra base de datos test con los siguientes valores:

```
fecha 920804700, valor 12345
fecha 920805000, valor 12357

etcétera.
```

Como ves, pueden introducirse más de un valor en la base de datos por ejecución del comando. Yo los agrupo de tres en tres para hacerlo legible, pero en realidad el máximo depende del sistema de operación.

Ahora podemos recuperar los datos usando ‘‘rrdtool fetch’’:

```
rrdtool fetch test.rrd AVERAGE --start 920804400 --end 920809200
```

Debes obtener esto como salida:

```
                   speed
```

```
920804400:         NaN
920804700:         NaN
920805000: 4.0000000000e-02
920805300: 2.0000000000e-02
920805600: 0.0000000000e+00
920805900: 0.0000000000e+00
920806200: 3.3333333333e-02
920806500: 3.3333333333e-02
920806800: 3.3333333333e-02
920807100: 2.0000000000e-02
920807400: 2.0000000000e-02
920807700: 2.0000000000e-02
920808000: 1.3333333333e-02
920808300: 1.6666666667e-02
920808600: 6.6666666667e-03
920808900: 3.3333333333e-03
920809200:         NaN
```

Si no, hay algo mal. Probablemente tu sistema de operación muestre ''NaN'' de otra forma; representa ''Not a Number'', o sea ''No es un número''. Si aparece ''U'' o ''UNKN'' o algo parecido, es lo mismo. Si hay alguna otra diferencia, probablemente te equivocaste al introducir algún P valor (asumiendo que mi tutorial está bien, por supuesto :–). En ese caso, borra la base de datos y prueba de nuevo.

Lo que representa exactamente esta salida lo vamos más adelante en el tutorial.

Hora de hacer algunos gráficos

Prueba este comando:

```
rrdtool graph speed.gif                                 \
        --start 920804400 --end 920808000               \
        DEF:myspeed=test.rrd:speed:AVERAGE              \
        LINE2:myspeed#FF0000
```

Este comando crea speed.gif, un gráfico de los datos desde las 12:00 hasta las 13:00. Contiene una definición de la variable myspeed y define el color como rojo. Notarás que el gráfico no comienza exactamente a las 12:00 sino a las 12:05, y es porque no tenemos datos suficientes como para calcular el promedio de velocidad antes de ese momento. Esto sólo ocurre en caso de que se pierdan algún muestreo, lo que esperamos que no debe ocurrir muy a menudo.

Si ha funcionado, ¡felicitaciones!. Si no, revisa qué puede estar mal.

La definición de colores se construye a partir del rojo, verde y azul. Especificas cuanto de cada uno de estos componentes vas a usar en hexadecimal: 00 significa ''nada de este color'' y FF significa ''este color a máxima intensidad''. El ''color'' blanco es la mezcla del rojo, verde y azul a toda intensidad: FFFFFF; el negro es la ausencia de todos los colores: 000000.

```
rojo    #FF0000
verde   #00FF00
azul    #0000FF
violeta #FF00FF     (mezcla de rojo y azul)
gris    #555555     (un tercio de cada uno de los colores)
```

El archivo GIF que acabas de crear puede verse con tu visor de archivos de imagen favorito. Los navegadores lo mostrarán usando la URL ''file://el/camino/de/directorios/hasta/speed.gif''

Gráficos con un poco de matemática

Cuando veas la imagen, notarás que el eje horizontal tiene unas etiquetas marcando las 12:10, 12:20, 12:30, 12:40 y 12:50. Los otros dos momentos (12:00 y 13:00) no se pueden mostrar bien por falta de datos, así que el programa se los salta. El eje vertical muestra el rango de los valores que entramos. Introdujimos los kilómetros y luego dividimos entre 300 segundos, por lo que obtuvimos valores bastante bajos. Para ser exactos, el primer valor, 12 (12357−12345), dividido entre 300 da 0.04, lo que RRDtool muestra como ‘‘40m’’, o sea ‘‘40/1000’’. ¡La ‘‘m’’' no tiene nada que ver con metros, kilómetros o milímetros!. RRDtool no sabe nada de unidades, el sólo trabaja con números, no con metros.

Donde nos equivocamos fue en que debimos medir en metros. Así, (12357000−12345000)/300 = 12000/300 = 40.

Vamos a corregirlo. Podríamos recrear la base de datos con los valores correctos, pero hay una forma mejor: ¡haciendo los cálculos mientras creamos el archivo gif!

```
rrdtool graph speed2.gif                              \
   --start 920804400 --end 920808000                  \
   --vertical-label m/s                               \
   DEF:myspeed=test.rrd:speed:AVERAGE                 \
   CDEF:realspeed=myspeed,1000,*                      \
   LINE2:realspeed#FF0000
```

Cuando veas esta imagen, notarás que la ‘‘m’’ ha desaparecido, y ahora tienes los resultados correctos. Además hemos añadido una etiqueta a la imagen. Apartando esto, el archivo GIF es el mismo.

Las operaciones están en la sección del CDEF y están escritas en Notación Polaca Inversa (Reverse Polish Notation o ‘‘RPN’’). En palabras, dice: ‘‘toma la fuente de datos myspeed y el numero 1000, y multiplícalos’’. No te molestes en meterte con RPN todavía, la veremos con más detalle más adelante. Además, puede que quieras leer mi tutorial sobre los CDEF y el tutorial de Steve Rader sobre RPN, pero primero terminemos con este.

¡Un momento! Si podemos multiplicar los valores por mil, entonces, ¡también debería ser posible el mostrar la velocidad en kilómetros por hora usando los mismos datos!

Para cambiar el valor que medimos en metros por segundo, calculamos los metros por hora (valor * 3600) y dividimos entre 1000 para sacar los kilómetros por hora. Todo junto hace valor * (3600/1000) == valor * 3.6.

Como en nuestra base de datos cometimos un error guardando los valores en kilómetros, debemos compensar por ello, multiplicando por 100, por lo que al aplicar esta corrección nos queda valor * 3600.

Ahora vamos a crear este gif, agreándole un poco más de magia...

```
rrdtool graph speed3.gif                              \
   --start 920804400 --end 920808000                  \
   --vertical-label km/h                              \
   DEF:myspeed=test.rrd:speed:AVERAGE                 \
   "CDEF:kmh=myspeed,3600,*"                          \
   CDEF:fast=kmh,100,GT,kmh,0,IF                      \
   CDEF:good=kmh,100,GT,0,kmh,IF                      \
   HRULE:100#0000FF:"Maximum allowed"                 \
   AREA:good#00FF00:"Good speed"                      \
   AREA:fast#FF0000:"Too fast"
```

Esto luce mucho mejor. La velocidad en KM/H, y además tenemos una línea extra mostrando la velocidad máxima permitida (en el camino por donde conduzco). También le cambie los colores de la velocidad, y ahora paso de ser una línea a un área.

Los cálculos son más complejos ahora. Para calcular la velocidad ''aceptable'':

```
Verifica si la velocidad en kmh es mayor que 100      ( kmh,100 ) GT
Si es así, retorna 0, si no, retorna la velocidad    ((( kmh,100 ) GT ), 0, kmh)
```

Para calcular la parte de velocidad ''excesiva'':

```
Verifica si la velocidad en kmh es mayor que 100      ( kmh,100 ) GT
Si es así, retorna la velocidad, si no, retorna 0    ((( kmh,100) GT ), kmh, 0) I
```

Magia gráfica

Me gusta creer que virtualmente no hay limites para lo que RRDtool puede hacer con los datos. No voy a explicarlo en detalle, pero mira este GIF:

```
rrdtool graph speed4.gif                            \
   --start 920804400 --end 920808000                \
   --vertical-label km/h                            \
   DEF:myspeed=test.rrd:speed:AVERAGE               \
   "CDEF:kmh=myspeed,3600,*"                        \
   CDEF:fast=kmh,100,GT,100,0,IF                    \
   CDEF:over=kmh,100,GT,kmh,100,-,0,IF              \
   CDEF:good=kmh,100,GT,0,kmh,IF                    \
   HRULE:100#0000FF:"Maximum allowed"               \
   AREA:good#00FF00:"Good speed"                    \
   AREA:fast#550000:"Too fast"                      \
   STACK:over#FF0000:"Over speed"
```

Vamos a crear una página HTML simple para ver los tres archivos GIF:

```
<HTML><HEAD><TITLE>Velocidad</TITLE></HEAD><BODY>
<IMG src="speed2.gif" alt="Speed in meters per second">
<BR>
<IMG src="speed3.gif" alt="Speed in kilometers per hour">
<BR>
<IMG src="speed4.gif" alt="Traveled too fast?">
</BODY></HTML>
```

Guárdalo como ''speed.html'' o algo parecido, y examínalo con un navegador.

Ahora, todo lo que tienes que hacer es medir los datos regularmente y actualizar la base de datos. Cuando quieras verlos, vuelve a crear los archivos GIF y asegúrate que se carguen de nuevo en tu navegador (Nota: presionar el botón de ''refrescar'' puede no ser suficiente; en particular, Netscape tiene un problema al respecto, por lo que necesitaras darle al botón mientras presionas la tecla de mayúsculas.

Actualizaciones de verdad

Ya hemos usado el comando ''update''; vimos que recibia uno o más parámetros en el formato: ''<fecha>:<valor>''. Para facilitarte las cosas, puedes obtener la fecha actual colocando ''N'' en la fecha. También podrías usar la función ''time'' de Perl para obtenerla. El ejemplo más corto de todo el tutorial :)

```
perl -e 'print time, "\n" '
```

Ahora, la forma de poner a correr un programa a intervalos regulares de tiempo depende del sistema de operación. La actualización, en pseudo-código, sería:

```
Toma el valor, colócalo en la variable "$speed"
rrdtool update speed.rrd N:$speed
```

(Pero no lo hagas sobre nuestra base de datos de pruebas, que aún la vamos a usar en otros ejemplos.

Eso es todo. Ejecutando este script cada 5 minutos, lo único que tienes que hacer para ver los gráficos actuales es correr los ejemplos anteriores, que también puedes poner en un script. Luego de correrlo, basta con cargar index.html

Unas palabras sobre SNMP

Me imagino que muy pocas personas serán capaces de obtener en su ordenador datos reales de su coche cada 5 minutos; los demás nos tendremos que conformar con algún otro contador. Puedes, por ejemplo, medir la cantidad de páginas que ha hecho una impresora, cuanto café has hecho con la cafetera, el medidor del consumo de electricidad, o cualquier otra cosa. Cualquier contador incremental puede monitorizarse y graficarse con lo que has aprendido hasta ahora. Más adelante, veremos también como monitorizar otro tipo de valores, como la temperatura. La mayoría usaremos alguna vez un contador que lleve la cuenta de cuantos octetos (bytes) a transferido un dispositivo de red, así que vamos a ver como hacer esto. Empezaremos describiendo como recoger los datos. Hay quien dirá que hay herramientas que pueden recoger estos datos por ti. ¡Es cierto! Pero, creo que es importante darse cuenta de que no son necesarias. Cuando tienes que determinar porqué algo no funciona, necesitas saber cómo funciona en primer lugar.

Una herramienta que mencionamos brevemente al principio del documento es SNMP. SNMP es una forma de comunicarse con tus equipos. La herramienta particular que voy a usar más adelante se llama ''snmpget'', y funciona así:

```
snmpget dispositivo clave OID
```

En ''dispositivo'' colocas el nombre o dirección IP del equipo a monitorizar. En clave, colocas la ''cadena de caracteres de la comunidad de lectura'', como se le denomina en el mundillo SNMP. Muchos dispositivos aceptarán ''public'' como cadena por defecto, pero por razones de privacidad y seguridad esta clave puede estar deshabilitada. Consulta la documentación correspondiente al dispositivo o programa.

Luego esta el tercer parámetro, llamado OID (Object IDentifier, identificador de objeto).

Al principio, cuando empiezas a aprender sobre SNMP, parece muy confuso. No lo es tanto cuando le hechas una ojeada a los ''MIB'' (Manager Information Base, o Base de Información Administrativa). Es un árbol invertido que describe los datos, empezando en un nodo raíz desde el que parten varias ramas. Cada rama termina en otro nodo y puede abrir nuevas sub-ramas. Cada rama tiene un nombre, y forman un camino que nos lleva hasta el fondo del árbol. En este ejemplo, las ramas que vamos a tomar se llaman iso, org, dod, internet, mgmt y mib-2. También pueden accederse por su número relativo; en este caso, estos números son 1, 3, 6, 1, 2 y 1:

```
iso.org.dod.internet.mgmt.mib-2 (1.3.6.1.2.1)
```

En algunos programas se usa un punto al iniciar el OID. Esto puede ser confuso; no hay ningún punto inicial en la especificación de los OID... sin embargo, algunos programas usan por defecto un prefijo inicial. Para indicar la diferencia entre los OID abreviados (o sea, a los que se le pondrá el prefijo inicial) y los completos, estos programas necesitan que los OID completos empiecen por un punto. Para empeorar las cosas, se usan varios prefijos distintos...

De acuerdo, sigamos con el inicio de nuestro OID: teníamos 1.3.6.1.2.1 . Ahora, nos interesa la rama ''interfaces'', que tiene el número dos (o sea, 1.3.6.1.2.1.2, o 1.3.6.1.2.1.interfaces).

Lo primero es hacernos con un programa SNMP. Busca algún paquete pre-compilado para tu plataforma, si no, puedes buscar el código fuente y compilarlo tu mismo. En Internet encontrarás muchos programas, búscalos con un motor de búsqueda o como prefieras. Mi sugerencia es que busques el paquete CMU-SNMP, que esta bastante difundido.

Asumamos que ya tienes el programa. Empecemos por tomar ciertos datos que están disponibles en la mayoría de los sistemas. Recuerda: hay un nombre abreviado para la parte del árbol que más nos interesa.

Voy a usar la versión corta, ya que creo que este documento ya es lo bastante largo. Si no te funciona, añádele el prefijo .1.3.6.1.2.1 y prueba de nuevo. O prueba leyendo el manual; sáltate las partes que no entiendas aún, y busca las secciones que hablan de como arrancar y usar el programa.

```
snmpget myrouter public system.sysdescr.0
```

El dispositivo deberá contestarte con una descripción, probablemente vacía, de sí mismo. Si no consigues una respuesta válida, prueba con otra ‘‘clave’’ u otro dispositivo; no podemos seguir hasta tener un resultado.

```
snmpget myrouter public interfaces.ifnumber.0
```

Con suerte, usando este comando obtendrás un número como resultado: el número de interfaces del dispositivo. Si es así, seguiremos adelante con otro programa, llamado ‘‘snmpwalk’’

```
snmpwalk myrouter public interfaces.iftable.ifentry.ifdescr
```

Si obtienes una lista de interfaces, ya casi hemos llegado. Aquí tienes un ejemplo del resultado:

```
[user@host /home/alex]$ snmpwalk cisco public 2.2.1.2
interfaces.ifTable.ifEntry.ifDescr.1 = "BRI0: B-Channel 1"
interfaces.ifTable.ifEntry.ifDescr.2 = "BRI0: B-Channel 2"
interfaces.ifTable.ifEntry.ifDescr.3 = "BRI0" Hex: 42 52 49 30
interfaces.ifTable.ifEntry.ifDescr.4 = "Ethernet0"
interfaces.ifTable.ifEntry.ifDescr.5 = "Loopback0"
```

En este equipo CISCO, quiero monitorizar la interfaz ‘‘Ethernet0’’. Viendo que es la cuarta, pruebo con:

```
[user@host /home/alex]$ snmpget cisco public 2.2.1.10.4 2.2.1.16.4

interfaces.ifTable.ifEntry.ifInOctets.4 = 2290729126
interfaces.ifTable.ifEntry.ifOutOctets.4 = 1256486519
```

Entonces, tengo 2 OIDs que monitorizar, y son (en el formato largo, ahora):

```
1.3.6.1.2.1.2.2.1.10

     y

1.3.6.1.2.1.2.2.1.16
```

, ambas con el número de interfaz de 4

No te engañes, esto no lo logre yo al primer intento. Me tomó un tiempo entender lo que significaban todos estos números; ayuda cuando se traducen en un texto descriptivo... por lo menos, cuando oigas hablar de MIBs y OIDs, ahora sabrás de qué se trata. No te olvides del número de interfaz (0 si el valor no depende de una interfaz), y prueba con snmpwalk si no obtienes una respuesta clara con snmpget.

Si entendiste todo esto, y obtienes resultados del dispositivo con el que estás probando, sigue adelante con el tutorial. Si no, vuelve a leer esta sección; es importante

Un ejemplo real

Ok, empecemos con la diversión. Primero, crea una base de datos nueva. Vamos a guardar en ella 2 contadores, ‘‘input’’ y ‘‘ouput’’. Los datos los vamos a guardar en archivos que los promediarán, tomando grupos de 1, 6, 24 o 288 muestras. También archivaremos los valores máximos. Lo explicaremos con más detalle después. El intervalo de tiempo entre las muestras será de 300 segundos (5 minutos).

```
1 muestra "promediada" sigue siendo 1 muestra cada 5 minutos
6 muestras promediadas son un promedio de cada 30 minutos
24 muestras promediadas son un promedio de cada 2 horas
288 muestras promediadas son un promedio de cada día
```

Vamos a tratar de ser compatibles con MRTG, que guarda más o menos esta cantidad de datos:

```
600 muestras de 5 minutos:          2 días y 2 horas
600 promedios de 30 minutos:        12.5 días
600 promedios de 2 horas:           50 días
600 promedios de 1 día:             732 días
```

Uniendo todos estos rangos tenemos que en total guardamos datos de unos 797 días. RRDtool guarda los datos de una forma distinta a MRTG; no empieza el archivo ‘‘semanal’’ donde acaba el ‘‘diario’’, sino que ambos archivos contienen la información más reciente, ¡por lo que con RRDtool archivamos más datos que con MRTG!

Necesitaremos:

```
600 muestras de 5 minutos    (2 días y 2 horas)
700 entradas de 30 minutos   (2 días y 2 horas, más 12.5 días)
775 entradas de 2 horas      (lo anterior + 50 días)
797 entradas de 1 día        (lo anterior + 732 días, redondeando)
```

```
  rrdtool create myrouter.rrd         \
           DS:input:COUNTER:600:U:U   \
           DS:output:COUNTER:600:U:U  \
           RRA:AVERAGE:0.5:1:600      \
           RRA:AVERAGE:0.5:6:700      \
           RRA:AVERAGE:0.5:24:775     \
           RRA:AVERAGE:0.5:288:797    \
           RRA:MAX:0.5:1:600          \
           RRA:MAX:0.5:6:700          \
           RRA:MAX:0.5:24:775         \
           RRA:MAX:0.5:288:797
```

Lo siguiente es recoger los datos y guardarlos, como en el ejemplo siguiente. Esta parcialmente en pseudocódigo, por lo que tendrás que buscar exactamente como hacerlo funcionar en tu sistema operativo.

```
mientras no sea el fin del universo
hacer
   tomar el resultado de
       snmpget router community 2.2.1.10.4
   en la variable $in
   tomar el resultado de
       snmpget router community 2.2.1.16.4
   en la variable $out
   rrdtool update myrouter.rrd N:$in:$out
   esperar 5 minutos
hecho
```

Luego, tras recoger datos por un día, crea una imagen, usando:

```
rrdtool graph myrouter-day.gif --start -86400 \
         DEF:inoctets=myrouter.rrd:input:AVERAGE \
         DEF:outoctets=myrouter.rrd:output:AVERAGE \
         AREA:inoctets#00FF00:"In traffic" \
         LINE1:outoctets#0000FF:"Out traffic"
```

Este comando debe producir un gráfico del tráfico del día. Un día son 24 horas, de 60 minutos, de 60 segundos: 24*60*60=86400, o sea que empezamos a ‘‘ahora’’ menos 86400 segundos. Definimos (con los DEFs) ‘‘inoctets’’ y ‘‘outoctets’’ como los valores promedio de la base da datos myrouter.rrd, dibujando un área para el tráfico de entrada y una línea para el tráfico de salida.

Mira la imagen y sigue recogiendo datos por unos cuantos días. Si lo deseas, puedes probar con los ejemplos de la base de datos de pruebas y ver si puedes hacer trabajar las diversas opciones y operaciones.

Sugerencia:

Haz un gráfico que muestre el tráfico en bytes por segundo y en bits por segundo. Colorea el tráfico Ethernet rojo si sobrepasa los cuatro megabits por segundo.

Funciones de consolidación

Unos cuantos párrafos atrás hablábamos sobre la posibilidad de guardar el valor máximo en vez del promedio. Profundicemos un poco en este tema.

Recordemos lo que hablábamos sobre la velocidad de un coche. Supongamos que manejamos a 144 KM/H durante 5 minutos y luego nos detiene la policía durante unos 25 minutos. Al finalizar el regaño, tomamos nuestro portátil y creamos una imagen desde nuestra base de datos. Si visualizamos la segunda RRA que creamos, tendremos el promedio de 6 muestreos. Las velocidades registradas serian 144+0+0+0+0+0=144, lo que en promedio nos da una velocidad de 24 KM/H., con lo que nos igual nos pondrían una multa, sólo que no por exceso de velocidad.

Obviamente, en este caso, no deberíamos tomar en cuenta los promedios. Estos son útiles en varios casos. Por ejemplo, si queremos ver cuantos KM hemos viajado, este sería el gráfico más indicado. Pero por otro lado, para ver la velocidad ha la que hemos viajado, los valores máximos son más adecuados.

Es lo mismo con los datos que recogemos. Si quieres saber la cantidad total, mira los promedios. Si quieres ver la velocidad, mira los máximos. Con el tiempo, ambas cantidades se separan cada vez más. En la última base de datos que creamos, había dos archivos que guardaban los datos de cada día. El archivo que guarda los promedios mostrará valores bajos, mientras que el de máximos mostrará valores más altos. Para mi coche, mostraría valores promedio de 96/24=4 KM/H (viajo unos 96 kilómetros por día), y máximos de 1220 KM/H (la velocidad máxima que alcanzo cada día)

Como ves, una gran diferencia. No mires el segundo gráfico para estimar la distancia que recorro, ni al primero para estimar la velocidad a la que voy. Esto sólo funciona con muestras muy cercanas, pero no si sacas promedios.

Algunas veces, hago un viaje largo. Si hago un recorrido por Europa, conduciendo por unas 12 horas, el primer gráfico subirá a unos 60 KM/H. El segundo mostrará unos 180 KM/H. Esto significa que recorrí unos 60 KM/H por 24 horas = 1440 KM. Muestra además que fui a una velocidad promedio mayor a la normal y a un máximo de 180 KM/H, ¡no que fui 8 horas a una velocidad fija de 180 KM/H! Este es un ejemplo real: tengo que seguir la corriente en las autopistas de Alemania, detenerme por gasolina y café de vez en cuando, manejar más lentamente por Austria y Holanda, e ir con cuidado en las montañas y las villas. Si viéramos los gráficos de los promedios de cada 5 minutos, la imagen sería completamente distinta; veríamos los mismos valores de promedio y de máxima. (suponiendo que las mediciones fueran cada 300 segundos). Se podría ver cuando paré, cuando iba en primera, cuando iba por las autopistas, etc. La granularidad de los datos es más alta, por lo que se tiene más información. Sin embargo, esto nos lleva unas 12 muestras por hora, o 288 al día, lo cual es mucho para guardar por un periodo de tiempo largo. Por lo tanto, sacamos el promedio, guardando eventualmente un solo valor por día. Con este único valor, no podemos ver mucho.

Es importante comprender lo que expuesto en estos últimos párrafos. Unos ejes y unas líneas no tienen ningún valor por si mismos; hay que saber que representan e interpretar correctamente los valores obtenidos. Sean cuales sean los datos, esto siempre será cierto.

El mayor error que puedes cometer es usar los datos recogidos para algo para lo cual no sirven. En ese caso, seria hasta mejor no tener gráfico alguno.

Repasemos lo que sabemos

Ahora ya sabes como crear una base de datos. Puedes guardar valores en ella, extraerlos creando un gráfico, hacer operaciones matemáticas con ellos desde la base de datos y visualizar los resultados de estas en vez de los datos originales. Vimos la diferencia entre los promedios y los máximos y cuando debemos usar cada uno (o al menos una idea de ello)

RRDtool puede hacer más de lo que hemos visto hasta ahora. Pero antes de continuar, te recomiendo que releas el texto desde el principio y pruebes a hacerle algunas modificaciones a los ejemplos. Asegúrate de entenderlo todo. El esfuerzo valdrá la pena, y te ayudará, no sólo con el resto del documento, sino en tu trabajo diario de monitorización, mucho después de terminar con esta introducción.

Tipos de fuentes de datos

De acuerdo, quieres continuar. Bienvenido de vuelta otra vez y prepárate; voy a ir más rápido con los ejemplos y explicaciones.

Ya vimos que, para ver el cambio de un contador a lo largo del tiempo, tenemos que tomar dos números y dividir la diferencia entre el tiempo transcurrido entre las mediciones. Para los ejemplos que hemos visto es lo lógico, pero hay otras posibilidades. Por ejemplo, mi enrutador me puede dar la temperatura actual en tres puntos distintos, la entrada de aire, el llamado ‘‘punto caliente’’ y la salida de ventilación. Estos valores no son contadores; si tomo los valores de dos muestreos y lo divido entre 300 segundos, obtendré el cambio de temperatura por segundo. ¡Esperemos que sea cero, o tendríamos un incendio en el cuarto de ordenadores! :)

Entonces, ¿que hacemos? Podemos decirle a RRDtool que guarde los valores tal como los medimos (esto no es exactamente así, pero se aproxima bastante a la verdad). Así, los gráficos se verán mucho mejor. Puedo ver cuando el enrutador está trabajando más (en serio, funciona; como usa más electricidad, genera más calor y sube la temperatura), puedo saber cuando me he dejado las puertas abiertas (el cuarto de ordenadores tiene aire acondicionado; con las puertas abiertas el aire caliente del resto del edificion entra y sube la temperatura en la entrada de aire del enrutador), etc. Antes usamos un tipo de datos de ‘‘contador’’, ahora usaremos un tipo de datos diferente, con un nombre diferente, GAUGE. Tenemos otros tipos:

```
- COUNTER este ya lo conocemos
- GAUGE   este acabamos de verlo
- DERIVE
- ABSOLUTE
```

Los otros dos tipos son DERIVE y ABSOLUTE. ABSOLUTE puede usarse igual que COUNTER, con una diferencia; RRDtool asume que el contador se reinicia cada vez que se lee. O en otras palabras; el delta entre los valores no hay que calcularlo, mientras que con COUNTER RRDtool tiene que sacar él la cuenta. Por ejemplo, nuestro primer ejemplo, (12345, 12357, 12363, 12363), sería (unknown, 12, 6, 0) en ABSOLUTE. El otro tipo, DERIVE, es como COUNTER, pero al contrario de COUNTER, este valor también puede decrecer, por lo que puede tenerse un delta negativo.

Vamos a probarlos todos:

```
rrdtool create all.rrd --start 978300900 \
         DS:a:COUNTER:600:U:U \
         DS:b:GAUGE:600:U:U \
         DS:c:DERIVE:600:U:U \
         DS:d:ABSOLUTE:600:U:U \
         RRA:AVERAGE:0.5:1:10
rrdtool update all.rrd \
         978301200:300:1:600:300    \
         978301500:600:3:1200:600   \
         978301800:900:5:1800:900   \
         978302100:1200:3:2400:1200 \
         978302400:1500:1:2400:1500 \
         978302700:1800:2:1800:1800 \
         978303000:2100:4:0:2100    \
         978303300:2400:6:600:2400  \
         978303600:2700:4:600:2700  \
         978303900:3000:2:1200:3000
rrdtool graph all1.gif -s 978300600 -e 978304200 -h 400 \
         DEF:linea=all.rrd:a:AVERAGE LINE3:linea#FF0000:"Line A" \
         DEF:lineb=all.rrd:b:AVERAGE LINE3:lineb#00FF00:"Line B" \
         DEF:linec=all.rrd:c:AVERAGE LINE3:linec#0000FF:"Line C" \
         DEF:lined=all.rrd:d:AVERAGE LINE3:lined#000000:"Line D"
```

RRDtool bajo el microscopio

- La línea A es un contador, por lo que debe incrementarse continuamente y RRDtool tiene que calcular las diferencias. Además RRDtool tiene que dividir la diferencia entre el tiempo transcurrido. Esto debería terminar con una línea recta en 1 (los deltas son 300, y los intervalos son de 300)
- La línea B es de tipo GAUGE. Estos son los valores ''reales'', así que el gráfico debe mostrar lo mismo que los valores que introducimos: una especie de onda
- La línea C es de tipo DERIVE. Es un contador, y puede decrecer. Va entre 2400 y 0, con 1800 en el medio.
- La línea D es de tipo ABSOLUTE. Esto es, es un contador pero no hay que calcular las diferencias. Los números son iguales a la línea A, y espero que puedas ver la diferencia en los gráficos.

Esto equivale a los valores siguientes, empezando a las 23:10 y terminando a las 00:10 (las U significan desconocido).

```
 - Línea  A:  u  u  1  1  1  1  1  1  1  1  1  u
 - Línea  B:  u  1  3  5  3  1  2  4  6  4  2  u
 - Línea  C:  u  u  2  2  2  0 -2 -6  2  0  2  u
 - Línea  D:  u  1  2  3  4  5  6  7  8  9 10  u
```

Si tu archivo GIF muestra todo esto, has entrado los datos correctamente, tu programa RRDtool está funcionando bien, el visor de gráficos no te engaña y hemos entrado en el 2000 sin problemas :) Puedes probar el mismo ejemplo cuatro veces, una por cada línea.

Revisemos los datos otra vez:

- Línea A: 300, 600, 900 , etc. La diferencia del contador es siempre 300, igual que el intervalo de tiempo transcurrido entre mediciones. Por lo tanto, el promedio siempre es 1. Pero, ¿por qué el primer punto tiene un valor de ‘‘desconocido’’? ¿Acaso no era conocido el valor que pusimos en la base de datos? ¡Si! Pero no teníamos un valor inicial para calcular la diferencia. Sería un error asumir que el contador empezaba en 0, así que no conocemos el valor de la diferencia
- Línea B: No hay nada que calcular, los valores son los mismos que se introdujeron en la base de datos.
- Línea C: De nuevo, no conocemos el valor inicial antes de la primera medición, así que se aplica el mismo razonamiento que para la línea A. En este caso las diferencias no son constantes, así que la línea no es recta. Si hubiésemos puesto los mismos valores que en la línea A, el gráfico sería el mismo. Al contrario que COUNTER, el valor puede decrecer, y espero mostrarte más adelante el por que de la diferencia entre ambos tipos.
- Línea D: En este caso, el dispositivo nos da las diferencias por sí mismo. Por lo tanto, conocemos la diferencia inicial, y podemos graficarla. Tenemos los mismos valores que en la línea A, pero su significado es distinto, por lo que el gráfico también lo es. En este caso, las diferencias se incrementan en 300 cada vez, mientras que el intervalo de tiempo permanece constante en 300 segundos, por lo que la división nos da resultados cada vez mayores.

Reinicialización de los contadores

Todavía nos quedan algunas cosas por ver. Nos quedan algunas opciones importantes por cubrir, y aun no hemos hablado de la reinicialización de contadores. Empecemos por ahí: Estamos en nuestro coche, vemos el contador y muestra 999987. Andamos unos 20 KM, así que el contador debe subir a 1000007. Desafortunadamente, el contador sólo tiene 6 dígitos, así que en realidad nos muestra 000007. Si estuviéramos guardando los valores en un tipo DERIVE, esto significaría que el contador retrocedió unos 999980 KM. Por supuesto esto no es cierto, por lo que necesitamos alguna protección contra estos casos. Esta protección sólo la tenemos para el tipo COUNTER, el cual de todas formas era el que teníamos que haber usado para este tipo de contador. ¿Cómo funciona? Los valores tipo COUNTER no deben decrecer nunca, ¡por lo que RRDtool asume en ese caso que el contador se ha reinicializado! Si la diferencia es negativa, esto se compensa sumando el valor máximo del contador + 1. Para nuestro coche, tendríamos:

```
Delta = 7 - 999987 = -999980    (en vez de 1000007-999987=20)

Delta real= -999980 + 999999 + 1 = 20
```

Al momento de escribir este documento, RRDtool maneja contadores de 32 o 64 bits de tamaño. Estos contadores pueden manejar los siguientes valores:

```
- 32 bits: 0 ..           4294967295
- 64 bits: 0 .. 18446744073709551615
```

Si estos valores te parecen raros, podemos verlos en formato hexadecimal:

```
- 32 bits: 0 ..         FFFFFFFF
- 64 bits: 0 .. FFFFFFFFFFFFFFFF
```

RRDtool maneja ambos contadores de la misma manera. Si ocurre un desbordamiento y la diferencia es negativa, RRDtool le suma primero el máximo del contador ‘‘menor’’ (32 bits) + 1 a la diferencia. Si aún así la diferencia es negativa, entonces el contador reinicializado era mayor (64 bits), por lo que se le suma el valor máximo del contador ‘‘largo’’ + 1 y se le resta el máximo del contador ‘‘pequeño’’ que sumamos erróneamente. Hay un problema con esto: supongamos que un contador largo se ha reinicializado al sumársele una diferencia muy grande; entonces es posible que al añadir el valor máximo del contador pequeño la diferencia nos dé positivo. En este caso poco probable, los valores resultantes no serian correctos. Para que ocurra esto, el incremento tiene que ser casi tan grande como el valor máximo del contador, por lo que de ocurrir es muy probable que halla varios problemas más en la configuración y no merezca la pena preocuparse sólo por este. Aún así, he incluido un ejemplo de este caso para que lo puedas juzgar por ti mismo.

A continuación, unos ejemplos de reinicialización de los contadores. Prueba de hacer los cálculos por ti mismo, o acepta mis resultados si tu calculadora no puede con los números :)

Números de corrección:

```
- 32 bits: (4294967295+1) =                                4294967296
- 64 bits: (18446744073709551615+1)-correction1 = 18446744069414584320

Antes:          4294967200
Incremento:            100
Debería ser:    4294967300
Pero es:                 4
Diferencia:    -4294967196
Corrección #1: -4294967196 + 4294967296 = 100

Antes:          18446744073709551000
Incremento:                      800
Debería ser:    18446744073709551800
Pero es:                         184
Diferencia:    -18446744073709550816
Corrección #1: -18446744073709550816 +4294967296 = -18446744069414583520
Corrección #2: -18446744069414583520 +18446744069414584320 = 800

Antes:          18446744073709551615 ( valor máximo )
Incremento:     18446744069414584320 ( incremento absurdo,
Debería ser:    36893488143124135935   mínimo para que
Pero es:        18446744069414584319   funcione el ejemplo)
Diferencia:              -4294967296
Corrección #1:  -4294967296 + 4294967296 = 0 (positivo,
                                              por tanto no se hace
                                              la segunda corrección)

Antes:          18446744073709551615 ( valor máximo )
Incremento:     18446744069414584319
Debería ser:    36893488143124135934
Pero es:        18446744069414584318
Diferencia:              -4294967297
Corrección #1:  -4294967297 +4294967296 = -1
Corrección #2:  -1 +18446744069414584320 = 18446744069414584319
```

Como puede verse en los últimos ejemplos, necesitas unos valores bastante extraños para hacer que RRDtool falle (asumiendo que no tenga ningún error el programa, por supuesto), así que esto no debería ocurrir. Sin embargo, SNMP o cualquier otro método que uses de recogida de datos puede también reportar algún valor erróneo ocasionalmente. No podemos prevenir todos los errores, pero podemos tomar algunas medidas. El comando "create" de RRDtool tiene dos parámetros especialmente para esto, que definen los valores mínimo y máximo permitidos. Hasta ahora hemos usado "U", "desconocido". Si le pasas valores para uno o ambos parámetros y RRDtool recibe un valor fuera de esos límites, los ignorará. Para un termómetro en grados Celsius, el mínimo absoluto es −273. Para mi enrutador, puedo asumir que ese mínimo es mucho mayor, digamos que 10. La temperatura máxima la pondría en unos 80 grados; más alto y el aparato no funcionaría. Para mi coche, nunca esperaría obtener valores negativos, y tampoco esperaría valores mayores a 230. Cualquier otra cosa sería un error. Pero recuerda, lo contrario no es cierto: si los valores pasan este examen no quiere decir que sean los correctos. Siempre examina bien el gráfico si los valores parecen extraños.

Remuestreo de los datos

Hay una funcionalidad importante de RRDtool que no hemos explicado todavía: es virtualmente imposible recoger los datos y pasarselos a RRDtool a intervalos exactos de tiempo. Por tanto, RRDtool interpola los datos a los intervalos exactos. Si no sabes que significa esto o como se hace, he aquí la ayuda que necesitas:

Supongamos un contador se incremente exactamente en 1 cada segundo. Queremos medirlo cada 300 segundos, por lo que deberíamos tener valores separados exactamente en 300. Sin embargo, por varias circunstancias llegamos unos segundos tarde y el intervalo es 303. La diferencia será por tanto 303. Obviamente, RRDtool no debe colocar 303 en la base de datos y dar así la impresión de que el contador se incrementó 303 en 300 segundos. Aquí es donde RRDtool interpola: alterá el valor 303 al valor que tendría 3 segundos antes y guarda 300 en 300 segundos. Digamos que la próxima vez llegamos justo a tiempo; por tanto, el intervalo actual es 297 segundos, por lo que el contador debería ser 297. De nuevo, RRDtool altera el valor y guarda 300, como debe ser.

```
        en RRD                     en realidad
tiempo+000:   0 delta="U"    tiempo+000:   0 delta="U"
tiempo+300: 300 delta=300    tiempo+300: 300 delta=300
tiempo+600: 600 delta=300    tiempo+603: 603 delta=303
tiempo+900: 900 delta=300    tiempo+900: 900 delta=297
```

Creemos dos bases de datos idénticas. He escogido el rango de tiempo entre 920805000 y 920805900.

```
rrdtool create seconds1.rrd   \
   --start 920804700          \
   DS:seconds:COUNTER:600:U:U \
   RRA:AVERAGE:0.5:1:24

para Unix: cp seconds1.rrd seconds2.rrd
para DOS: copy seconds1.rrd seconds2.rrd
para VMS:  y yo que sé :)

rrdtool update seconds1.rrd \
   920805000:000 920805300:300 920805600:600 920805900:900
rrdtool update seconds2.rrd \
   920805000:000 920805300:300 920805603:603 920805900:900
```

```
rrdtool graph seconds1.gif                          \
   --start 920804700 --end 920806200                \
   --height 200                                     \
   --upper-limit 1.05 --lower-limit 0.95 --rigid \
   DEF:seconds=seconds1.rrd:seconds:AVERAGE         \
   CDEF:unknown=seconds,UN                          \
   LINE2:seconds#0000FF                             \
   AREA:unknown#FF0000
rrdtool graph seconds2.gif                          \
   --start 920804700 --end 920806200                \
   --height 200                                     \
   --upper-limit 1.05 --lower-limit 0.95 --rigid \
   DEF:seconds=seconds2.rrd:seconds:AVERAGE         \
   CDEF:unknown=seconds,UN                          \
   LINE2:seconds#0000FF                             \
   AREA:unknown#FF0000
```

Los dos gráficos debe ser iguales.

RESUMEN

Es hora de concluir este documento. Ahora debes conocer lo básico como para trabajar con RRDtool y leer la documentación. Aún hay mucho más por descubrir acerca de RRDtool, y le encontrarás; más y más usos para la herramienta. Con los ejemplos y la herramienta puedes crear fácilmente muchos gráficos; también puedes usar las interfaces disponibles.

LISTA DE CORREO

Recuerda subscribirte a la lista de correo. Aunque no contestes los correos que aparecen en ella, te servirá de ayuda a ti y a los demás. Mucho de lo que se sobre MRTG (y por tanto sobre RRDtool), lo aprendí tan sólo con leer la lista, sin escribir. No hay por que preguntar las preguntas básicas, que ya tienen su respuesta en la FAQ (¡léela!). Con miles de usuarios a lo largo del mundo, siempre hay preguntas que tu puedes responder con lo aprendido en este y otros documentos.

VER TAMBIÉN

Las páginas del manual de RRDtool

AUTOR

Espero que hayas disfrutado con los ejemplos y las descripciones. Si es así, ayuda a otros refiriéndolos a este documento cuando te hagan preguntas básicas. No sólo obtendrán la respuesta, sino que aprenderán muchas otras cosas.

Alex van den Bogaerdt <alex@ergens.op.het.net>