UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMATICA

Trabajo Final presentado para obtener el grado de Especialista en Ingeniería de Software

Título: Gestión de datos con RRDtool en ambientes HPC

Autor: Lic. Leopoldo José RIOS

Director: Dr. Fernando G. TINETTI

INDICE GENERAL

1. Introducción

La recopilación de datos en ambientes HPC (High Performance Computing, Cómputo de Alto Rendimiento), como ser la temperatura ambiente del centro de datos y el consumo de CPU y Memoria, suele ser una tarea simple de realizar. Sin embargo, no siempre resulta trivial el poder almacenar y procesar estos datos de una manera eficaz y sistemática.

RRDtool es el acrónimo de Round Robin Database Tool, un software para gestionar bases de datos de series de tiempo, que utiliza la técnica round-robin (buffer circular) para el almacenamiento de datos. Esta herramienta, es hoy por hoy una de las más difundidas y utilizadas en el monitoreo de recursos computacionales, específicamente para el ámbito HPC, y de manera general, en equipos servidores standalone [1]. Se desea enfocar su estudio como soporte para la creación, almacenamiento, y gestión de datos obtenidos en un ambiente HPC.

Sin duda, el hecho de poder medir las capacidades de los recursos computacionales con que cuenta una organización dedicada a la investigación, es determinante para quienes ejercen la dirección y el control del funcionamiento operativo. El uso de la herramienta RRDtool en la visualización de datos del uso de recursos del clúster en tiempo real, permite en un sentido de planificación estratégica, establecer agendas y prioridades durante un período de tiempo; y desde el punto de vista de control, el poder verificar si el trabajo planificado, es efectivamente realizado. Será posible establecer por ejemplo, un rango de tiempo específico, y obtener así resultados estadísticos de un investigador en particular en cuanto al uso del clúster.

El mundo HPC hace uso de RRDtool por distintos motivos que se explicarán, y en muchos casos, por ser el soporte de gestión de datos de una herramienta de monitoreo llamada Ganglia [2]. Ganglia es una distribución de software libre, orientada a sistemas en clúster de alto rendimiento, que~~,~~ registra datos obtenidos del monitoreo de recursos del sistema sin que ello derive en complicadas situaciones operativas, y permite su visualización en tiempo real desde consolas web remotas.

2. Objetivo del estudio.

El contexto de este trabajo es abordar la técnica específica de la herramienta y describir las funcionalidades de consolidación de datos que aporta a la hora de gestionar y visualizar información. Se compara con la herramienta de Bases de Datos MySQL, por ser representativa del ambiente Linux/Unix, como forma de resaltar las características relevantes de ambas soluciones para manejo de datos y cómo RRDtool está específicamente optimizada o aprovechada.

El ámbito de aplicación es un sistema de computación en clúster, un agrupamiento de computadoras organizadas con el objetivo de resolver múltiples operaciones numéricas de manera eficiente. Cuando los nodos de cálculo se encuentran en actividad y la herramienta RRDtool es configurada de manera apropiada, es posible medir y registrar el estado de variables del sistema, como ser uso de memoria y CPU, uso de la red de

interconexión, a los efectos de una posterior evaluación. Es común, evaluar por ejemplo, cómo mejorar el software en el uso de memorias intermedias (cachés) al detectar excesos en la comunicación de red; o en la gestión de bloques de multiplicación ante un escaso valor de caché hit (aciertos en uso de memoria intermedia). En muchos casos interesa saber cómo mide una determinada aplicación de cálculo a los efectos de poder mejorarla en un sentido de eficiencia.

3. Estado del Arte.

La herramienta RRDtool fue escrita inicialmente por Tobias Oetiker con contribuciones de muchas otras personas [3]. RRDtool está disponible bajo los términos de Licencia Pública General GNU V2 o posterior [4].

RRDtool se originó de MRTG [5], que empezó como un pequeño script para graficar el uso de la conexión de datos de una universidad a Internet. MRTG fue más tarde utilizado como una herramienta para graficar otras fuentes de datos, incluyendo la temperatura, la velocidad, la tensión, el número de impresiones y similares.

El valor de RRDtool se atribuye a la relativa facilidad en la registración y análisis de datos que se reúnen desde múltiples tipos de fuentes de datos (o DS, Data Source). La parte de análisis de datos de RRDtool se basa en la capacidad de generar rápidamente representaciones gráficas de los valores de los datos recogidos durante un periodo de tiempo definible.

En muchos casos se usa RRDtool para almacenar y procesar datos recogidos a través de SNMP, un protocolo de aplicación estándar muy utilizado en comunicación de datos. Los datos serán muy probablemente bytes (o bits) transferidos desde y hacia una red o un Host.

La ‘parte trasera’ de muchos programas de monitoreo populares como Ganglia, MRTG, Nagios [6], collectd [7], nmon [8], y otros se basan en RRDtool [9].

Problema de escala.

Si consideramos todos los hosts integrados al SNCAD [10] al 20-05-2014, de sus 17 centros de computación adheridos al sistema, nos podemos preguntar ¿cómo podemos hacer para reunir medidas de utilización del uso de la CPU, desde cada host cada 10 segundos, u otro período de tiempo? Suponiendo tener 340 hosts, el sistema de monitoreo debería sondear 340 hosts por segundo para lograr una resolución de 10 segundos y lograr esa métrica particular. También sería necesario definir cómo almacenar, graficar y presentar los datos de forma rápida y eficiente.

Los defensores de la herramienta Ganglia afirman que lo planteado en el párrafo anterior, es un problema de escala, y por lo tanto es el dominio del problema para el que esta herramienta fue diseñada: el supervisar y recolectar cantidades masivas de mediciones del sistema en tiempo real para grandes instalaciones.

Las instalaciones grandes son interesantes porque nos obligan a reinventar o al menos reevaluar problemas que los administradores de sistemas pensábamos ya habíamos resuelto. Se puede decir que la perspectiva de lanzar algún script Perl es totalmente diferente cuando son 340 la cantidad de hosts que están involucrados.

Como el número de máquinas se vuelve cada vez más numeroso, es más probable preocuparnos por la eficiencia del protocolo de sondeo, y estar menos propensos a interactuar directamente con cada máquina. Las instalaciones pequeñas, tienden a favorecer el uso de herramientas que reduzcan al mínimo la necesidad de configurar hosts individualmente.

Existe hoy día, una tendencia en definir configuraciones centralizadas a partir de un demonio (proceso) central que se encuentra en algún punto identificable de la red de datos, que determina sondeos periódicos para cada host a los efectos de poder registrar su estado. Estos sistemas son relativamente fáciles de usar puesto que el procedimiento se basa en la instalación del agente en cada host, y en la configuración del proceso de sondeo en el servidor central.

4. Desarrollo.

RRDtool es una herramienta estándar de la industria Open Source, para el registro de datos de alto rendimiento, incorpora un sistema de gráficos de datos de series de tiempo RRDtool se puede integrar fácilmente en los scripts de Shell, Perl, Python, Ruby, etc.

Se hace prácticamente imposible el poder recoger datos y registrarlos en una BD tradicional (relacional) en intervalos exactos, aspecto que se demuestra luego en un ejemplo. Por lo tanto, la técnica de RRDtool interpola los datos, y logra almacenarlos en intervalos exactos.

Recordamos, que los datos se almacenan en una base de datos con técnica round-robin (buffer circular) para permitir que el espacio de almacenamiento del sistema permanezca constante en el tiempo. La base de datos tendrá siempre la misma cantidad de secuencias a lo largo de su vida útil. Cuando llegan nuevos datos, en el conjunto de datos, el de registro más antiguo se elimina (técnica FIFO). De esta manera, el conjunto de datos no crece en tamaño y, por tanto, no requiere mantenimiento. Almacena y recupera datos de ellos.

En un archivo de Base de Datos Round Robin (RRD), se almacenan solamente números. Significa que tiene que ser capaz de medir algún valor en varios puntos en el tiempo y proporcionar esta información al RRDtool.

Supongamos que un contador se incrementa en uno exactamente cada segundo y deseamos medirlo en intervalos de 300 segundos. Esto implica que hay que obtener valores exactamente a 300 segundos de diferencia. Sin embargo, debido a diversas circunstancias operativas, puede que el intervalo sea de por ejemplo 303 segundos. El valor (delta) en el registro de la base de datos, también será 303 en ese caso. Obviamente, RRDtool no debe poner 303 como dato para hacer creer que el contador se

incrementó en 303 en 300 segundos. Aquí es donde RRDtool interpola: altera el valor 303 como si hubiera sido almacenado anteriormente y será 300 en 300 segundos. Así, la siguiente registración y lectura se tomará al momento justo.

A partir de este principio, será posible vincular un determinado programa que se encuentre corriendo en un nodo de cálculo, y relacionar el estado de variables como caché hit en un determinado momento. El hecho de poder asegurar las lecturas de datos en los tiempos justos, es importante para diversos análisis.

Estas características de RRDtool, serán contrastadas con un sistema de Bases de Datos relacional MySQL, mediante la ejecución de tareas de registración, a efectos de observar sus virtudes y ventajas.

4.1. La Metodología de RRDtool

Son tres los pasos básicos para crear la Base de Datos RRDtool, recoger datos y graficarlos [11].

1) Inicializar la base de datos. Crear la base de datos RRD y prepararla para recibir datos. Hay que decidir la cantidad de datos a conservar, con qué frecuencia los datos serán actualizados y qué tipo de datos se espera recoger.

2) Recoger los conjuntos de datos. Un mecanismo puede estar definido por ejemplo por una tarea 'cron', que se ejecutará para recopilar datos utilizando un script para incorporar esos datos periódicamente en la base de datos. Este es el paso que probablemente tomará más tiempo para trabajar correctamente.

3) Crear el gráfico. El último paso es tomar los datos de la base de datos RRD, hacer cálculos que puede resultar necesarios y crear el gráfico en tiempo real. Es posible, que este paso sea utilizado desde una tarea programada y se deba trasladar el gráfico a un directorio del servidor web para facilitar la visualización.

4.2. ¿Cómo funciona RRDtool?

Es posible analizar el funcionamiento del programa rrdtool, a partir de las siguientes etapas [12].

4.2.1. Adquisición de Datos

Al supervisar el estado de un sistema de computación, puede llegar a ser conveniente obtener datos en un intervalo de tiempo constante, sin embargo no siempre es posible reunirlos exactamente en el momento en que uno lo desea. Se podrá advertir que RRDtool permite actualizaciones del registro de cabecera en el momento en que sea necesario, dado que interpola automáticamente el valor de los datos (DS) en el último intervalo horario y escribe este valor interpolado en el registro. El valor original que ha suministrado se almacena y se tiene en cuenta al interpolar la siguiente entrada.

4.2.2. Consolidación de datos

Es posible registrar datos en un intervalo de, por ejemplo, un minuto, y estar interesados en conocer el desarrollo de esos datos con respecto al año pasado. Lo que se hace generalmente, es almacenar los datos en intervalos de un minuto durante todo el año. La consecuencia de esta actividad por medios convencionales, podría consumir un considerable espacio en disco y también mucho tiempo en su análisis, cuando por ejemplo, se desea crear un gráfico para representar datos de todo un año.

RRDtool ofrece una solución a este problema a través de su función de consolidación de datos (CF). En la creación de una base de datos Round Robin (RRD), se puede definir en qué intervalo debe ocurrir esta consolidación, y qué función de consolidación será utilizada en la construcción de los valores consolidados (un promedio, el valor mínimo, el valor máximo, el último valor). Es posible definir cualquier número de diferentes configuraciones de consolidación dentro de una RRD. Todos ellos serán mantenidos a medida que los nuevos datos se cargan en la RRD.

4.2.3. Archivos round robin

Los valores de datos de la misma configuración de consolidación se almacenan en archivos Round Robin (RR). Esta es una manera muy eficiente para almacenar datos durante un determinado período de tiempo, utilizando una cantidad determinada y constante de espacio de almacenamiento.

El sistema funciona de esta manera: Si por ejemplo deseamos almacenar 1000 valores en un intervalo de 5, RRDtool asignará espacio para esos 1000 valores de datos y para una zona de cabecera. En la cabecera se almacenará punteros con el valor del área de almacenamiento donde se escribió por última vez.

El uso de archivos RR garantiza que la base de datos no crece con el tiempo y que los datos de mayor edad se eliminan automáticamente. Mediante el uso de la función de consolidación, es posible mantener los datos durante un tiempo más largo, al tiempo que reduce gradualmente la resolución de los datos a lo largo del eje de tiempo.

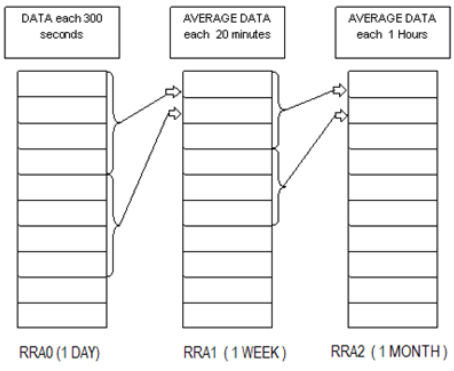


Fig-1

El uso de diferentes funciones de consolidación, permite almacenar exactamente el tipo de información que realmente nos interesa.

4.2.4. Datos desconocidos

Como ya se mencionó, a veces puede ocurrir que no hay datos disponibles al momento de la recolección cuando un valor se debe escribir a la RRD. La adquisición de datos no es posible por diferentes razones. RRDtool puede manejar estas situaciones mediante el almacenamiento de un valor ‘DESCONOCIDO’ en la base de

datos. El valor 'DESCONOCIDO' será reconocido y soportado en todas las funciones de la herramienta.

Al consolidar un conjunto de datos, la cantidad de valores de datos DESCONOCIDOS se contabiliza y cuando un nuevo valor consolidado está listo para ser escrito, se realiza primero una comprobación de validez para asegurarse de que el porcentaje de valores desconocidos en el punto de datos está por encima de un nivel configurable. Si no, un valor 'DESCONOCIDO' se escribirá en la RRD.

4.3. Primer ejemplo

Vamos a describir un primer ejemplo, diciendo que una de las más importantes funciones de RRDtool es la de generar informes en forma numérica y gráfica en base a los datos almacenados en uno o varios archivos de datos RRD. La característica gráfica es configurable en cuanto al tamaño, color y contenido. En el siguiente ejemplo se demostrarán las funcionalidades de esta herramienta.

Suponemos que nos encontramos conduciendo un automóvil. A las 12:05 leemos el cuentakilómetros y nos informa 12.345 kilómetros hasta ese momento. A las 12:10 lo miramos una vez más y se lee 12.357 kilómetros. Esto significa que ha recorrido 12 kilómetros en cinco minutos.

Es decir, viajamos 12 kilómetros que equivalen a 12.000 metros. Hicimos eso en cinco minutos o 300 segundos. La velocidad alcanzada es de 12000m/300s o 40m/s.

También es posible calcular la velocidad en km/h: 12 veces 5 minutos es una hora, así que tenemos que multiplicar 12 kilómetros por 12 para obtener 144 kmh.

Es necesario recordar que estos números son sólo promedios. No hay manera de averiguar a partir de ellos si se condujo a una velocidad constante.

4.3.1. Creación de la base de datos.

Consideramos un sistema Linux, con el siguiente script se crea la base de datos RRD:

```
leopoldo@linux-jos:~> cat crear_rdd.sh

#!/bin/sh

 rrdtool create test.rrd             \
   --start 1493992800  --step 300      \
            DS:speed:COUNTER:600:U:U   \
            RRA:AVERAGE:0.5:1:24       \
            RRA:AVERAGE:0.5:6:10
```

Una vez ejecutado el script, la herramienta crea la base de datos llamada 'test.rrd':

```
leopoldo@linux-jos:~> ls
```

```
total 8
drwxr-xr-x 1 leopoldo users   40 May 29 13:45 ./
drwxr-xr-x 1 leopoldo users  228 May 29 13:44 ../
-rwxr-xr-x 1 leopoldo users  208 May 29 13:45 crear_rdd.sh*
-rw-r--r-- 1 leopoldo users 1008 May 29 13:46 test.rrd
```

Veamos lo que se ha hecho:

Nuestra base de datos se genera el (1493992800), fecha en segundos basados en el sistema UNIX, se corresponde con la fecha y hora en que escribí este párrafo (29-05-2015 14:00:00).

Contiene una fuente de datos (DS-Data Source) llamada "speed" que representa a un contador. Este contador se lee cada cinco minutos (medida en segundos --step 300).

En la misma base de datos, dos archivos round robin (RRAS) se gestionan, el primero (RRA:AVERAGE:0.5:1:24) promedia los datos cada vez que se lee y mantiene 24 muestras reunidas cada 5 minutos, en total de 2 horas. El otro (RRA:AVERAGE:0.5:6:10) mantiene 10 promedios, calcula cada media hora 6 valores (cada 5 minutos).

4.3.2. Ingreso de datos a la base de datos.

Paso siguiente, se ingresan datos a la base de datos con algunos números. Al finalizar el procedimiento, vamos a pretender haber leído los siguientes números:

```
       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
```

La instrucción de ingreso es,

```
rrdtool update test.rrd 1493993100:12345 1493993400:12357
1493993700:12363
rrdtool update test.rrd 1493994000:12363 1493994300:12363
1493994700:12373
rrdtool update test.rrd 1493995000:12383 1493995300:12393
1493995700:12399
rrdtool update test.rrd 1493996000:12405 1493996300:12411
1493996700:12415
rrdtool update test.rrd 1493997000:12420 1493997300:12422
1493997700:12423
rrdtool update test.rrd 1493998000:
```

Es decir, el primer dato se lee a los 300 segundos del momento inicial, y se repite cada mismo tiempo a partir de 1493992800, a la vez en que el cuentakilómetros también aumenta:

```
time 1493993100, value 12345
time 1493993400, value 12357
time 1493993700, value 12363
```

Se puede apreciar la posibilidad de ingresar más de un valor en la base de datos en un solo comando. El verdadero valor máximo por línea depende del sistema operativo.

La recuperación de datos de nuestra base de datos se hace utilizando "rrdtool fetch", y si lo hacemos indicando que deseamos los datos tomados en un determinado período, nos queda:

```
rrdtool fetch test.rrd AVERAGE --start 1493993100 --end 1493998000
```

Lo cual nos debería traer:

```
                             speed

     1493993400: 4.0000000000e-02
     1493993700: 2.0000000000e-02
     1493994000: 0.0000000000e+00
     1493994300: 0.0000000000e+00
     1493994600: 2.5000000000e-02
     1493994900: 3.0555555556e-02
     1493995200: 3.3333333333e-02
     1493995500: 2.1111111111e-02
     1493995800: 1.6666666667e-02
     1493996100: 2.0000000000e-02
     1493996400: 1.3333333333e-02
     1493996700: 1.3333333333e-02
     1493997000: 1.6666666667e-02
     1493997300: 6.6666666667e-03
     1493997600: 2.5000000000e-03
     1493997900: 2.5000000000e-03
     1493998200: -nan
```

Además de ‘-nan’, también podemos llegar a encontrar "NaN" y eso en la mayoría de los casos, depende del sistema operativo. "NaN" significa "no es un número". Si su sistema operativo escribe en "U" o "UNKN" o algo similar también está bien.

4.3.3. Gráfico de datos

Ahora, y como el paso más importante que se desea mostrar, algo de magia: crear un gráfico. Vamos a ejecutar el siguiente script para generar el primer gráfico:

```
#!/bin/sh

 rrdtool graph speed1.png                                    \
         --start 1493993400 --end 1493998000                 \
         DEF:myspeed=test.rrd:speed:AVERAGE                  \
         LINE2:myspeed#FF0000
```

```
leopoldo@linux-jos: > ./grafico1.sh

481x141
```

Al mostrarlo en un navegador web, se observará el siguiente aspecto (Fig-2):

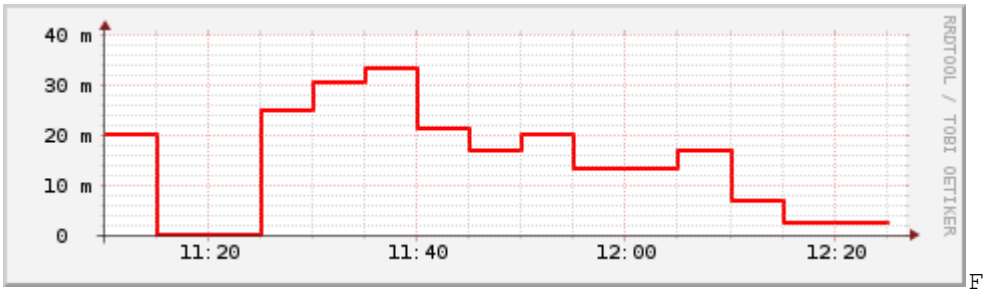


Fig-2

La serie de datos inicia a las 11:00 y finaliza a las 12:20. En el ejemplo hay una definición de una variable llamada ‘myspeed’, que utiliza datos de la fuente RRA "speed" de la base de datos "test.rrd". La línea trazada es de 2 píxeles de alto y representa la velocidad variable. El color es rojo, y está especificado por su valor rgb.

Los datos almacenados, pueden ser presentados en un gráfico con mucha más elaboración.

El eje vertical muestra el rango de tiempo definido, hemos proporcionado kilómetros y cuando se divide por 300 segundos, obviamente se obtienen números muy pequeños. Para ser exactos, el primer valor fue de 12 (12.357-12.345) y dividido por 300 esto da 0.04 que se muestra como "40 m", que significa "40/1000".

Si hubiéramos medido nuestras distancias en metros, esto habría sido (12357000-12345000) / 300 = 12000/300 = 40 metros. Dado que nos hacemos una mejor idea de los números en este rango, es posible corregirlo. Una forma sería volver a crear la base de datos y almacenar los datos correctos, pero afortunadamente RRDtool aporta funciones que permiten hacerlo mejor aún y un tanto más fácil, y es hacer algunos cálculos previos a crear el archivo gráfico png. Para hacerlo, vamos a correr el siguiente script:

```
leopoldo@linux-jos:> cat grafico2.sh
#!/bin/sh
   rrdtool graph speed2.png                                 \
      --start 1493993400 --end 1493998000                   \
      --vertical-label m/s                                  \
      DEF:myspeed=test.rrd:speed:AVERAGE                    \
      CDEF:realspeed=myspeed,1000,\*                        \
```

```
        LINE2:realspeed#FF0000
```

Que luego de la ejecución se obtiene el gráfico que se observa (Fig-3):

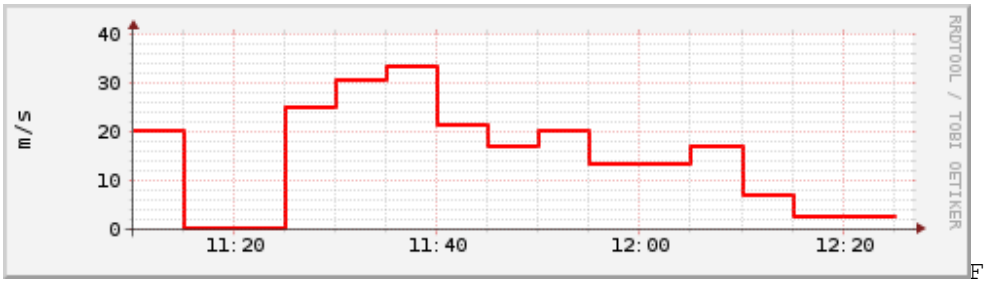


Fig-3

Del script anterior, se desprende la instrucción:

```
    "CDEF:realspeed=myspeed,1000,*"                    \
```

Estamos multiplicando el valor almacenado por 1000. Se escribe así entre comillas para que el sistema operativo no interprete una concatenación de caracteres.

Del gráfico de la Fig-3, se observa que se ha reemplazado la "m" (de mili) del gráfico anterior por “m/s”. En lo demás, el gráfico es el mismo.

Con el objeto de ampliar este último concepto, si podemos multiplicar los valores por 1000, también es posible visualizar, por ejemplo, kilómetros por hora a partir de los mismos datos.

Para cambiar un valor que se mide en metros por segundo hacemos:

```
    Calcular metros por hora:       valor * 3600
    Calcular kilómetros por hora:   valor / 1000
    Juntos esto hace: valor * (3600/1000) o el valor * 3.6
```

Para el caso de nuestra base de datos de ejemplo, la corrección sería:

```
    valor * 3.6  * 1000 == valor * 3600
```

Aplicando esta corrección, se puede apreciar algo más de magia:

```
leopoldo@linux-jos:> cat grafico3.sh
#!/bin/sh
rrdtool graph speed3.png                                \
      --start 1493993400 --end 1493998000               \
      --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:"Maxima permitida"              \
        AREA:good#00FF00:"Aceptable"                     \
        AREA:fast#FF0000:"Muy rapido"
```

Una vez ejecutado el script grafico3.sh, obtenemos la siguiente figura (Fig-4):

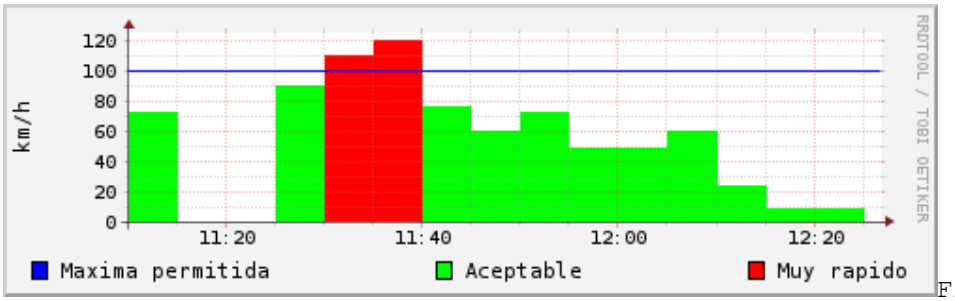


Fig-4

En este caso de la Fig-4, la velocidad se muestra en km/h, agrega una línea adicional con la velocidad máxima permitida en color azul, dada por la instrucción 'HRULE:100#0000FF:"Maxima permitida"'.

Para lograr las mediciones de velocidad dentro de los límites de velocidad hacemos:

```
      CDEF:fast=kmh,100,GT,kmh,0,IF
```

- Verifica si el valor de kmh es mayor que 100, si lo es, retorna '0'

Para valores superiores al límite de velocidad:

```
      CDEF:good=kmh,100,GT,0,kmh,IF
```

- Verifica si el valor de kmh es mayor que 100, si lo es, retorna 'kmh'

Con este ejemplo se demuestra la gran amplitud que posee la herramienta en la definición de cálculos con los datos almacenados. Podemos generar otro grafico para mejorar lo expresado:

```
leopoldo@linux-jos:> cat grafico4.sh
#!/bin/sh
   rrdtool graph speed4.png                          \
      --start 1493993400 --end 1493998000            \
      --vertical-label km/h                          \
      DEF:myspeed=test.rrd:speed:AVERAGE             \
      CDEF:nonans=myspeed,UN,0,myspeed,IF            \
      CDEF:kmh=nonans,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:"Maxima permitida"            \
      AREA:good#00FF00:"Aceptable"                  \
      AREA:fast#550000:"Muy rapido"                  \
      STACK:over#FF0000:"Exceso de velocidad"
```

Luego de la ejecución del script ‘grafico4.sh’, obtenemos (Fig-5):

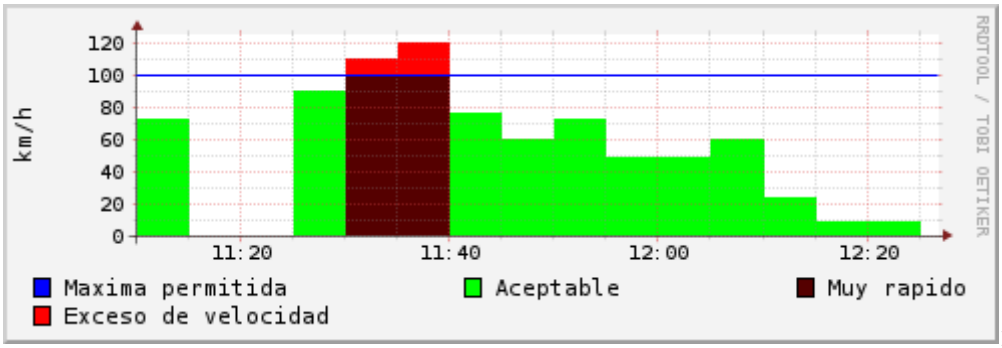


Fig-5

El área en color negro, representa los datos que llegan a una velocidad considerada como “Muy rápido”, hasta pasar el límite de velocidad permitido de 100 Km/h.

Los datos ‘DESCONOCIDOS’ se eliminan. El CDEF 'nonans' se agrega, y la sexta línea (que solía ser la quinta línea) se utiliza para leer 'CDEF: kmh = myspeed, 3600, *'.

4.4. RRDtool bajo la lupa

A partir de las descripciones básicas aportadas, sumadas a un primer ejemplo concretado, pasamos a un nivel de detalle más específico.

4.4.1. Recolección de datos

Es posible que el proceso de recolección de datos sea el más importante de toda esta mecánica. Para recolectar la información de los dispositivos y variables que deseamos medir, se establecen mecanismos, en algunos casos estandarizados mediante protocolos específicos como el protocolo SNMP (Simple Network Management Protocol). En sistemas Linux es posible desarrollar una estrategia de recolección basada en ‘cron y crontab’, procesos que regulan la ejecución de otros procedimientos planificados por el administrador del sistema.

Cron y crontab

El nombre cron viene del griego chronos que significa “tiempo”. En sistemas operativos Unix/Linux, cron es un demonio, cuya función es administrar procesos en segundo plano. En muchos casos, éstos ejecutan otros procesos o scripts a intervalos regulares de tiempo (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab. [13]

Se obtiene información sobre qué programas y cuándo deben ejecutarse a partir de la lista de entradas de crontab de usuarios y sistema. Sólo el usuario root tiene acceso a los crontab’s del sistema, mientras que cada usuario tiene acceso a sus propios crontab. Es posible definir a determinados usuarios en el uso de cron.

El demonio cron se inicia a partir de /etc/rc.d/ o /etc/init.d dependiendo de la distribución de Linux. Cron se ejecuta en segundo plano, revisa cada minuto la tabla de tareas crontab en /etc/crontab o en /var/spool/cron en búsqueda de tareas que se deban cumplir. Como usuario podemos agregar comandos o scripts con tareas a cron para automatizar algunos procesos. Esto es útil por ejemplo para automatizar la actualización de un sistema o un buen sistema de respaldos.

Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario. Crontab verificará la fecha y hora en que se debe ejecutar el script o el comando, los permisos de ejecución y lo realizará en background. Cada usuario puede tener su propio archivo crontab, de hecho el /etc/crontab se asume que es el archivo crontab del usuario root, cuando los usuarios normales (e incluso root) desean generar su propio archivo de crontab, entonces utilizaremos el comando crontab.

Ejemplo:

Ejecutar cada minuto de cada hora, el comando ping:

```
1 * * * *  /sbin/ping 127.0.0.1
```

Significados de cada campo

minutos	*hora*	*día-del-mes*	*mes*	*día-de-la-semana*	
*	*	*	*	*	*Comando a ejecutar*

Si deseamos ejecutar el comando 'ping' cada diez (10) minutos de cada hora, hacemos:

```
*/10 * * * *  /sbin/ping 127.0.0.1
```

Si se desea mostrar un mensaje, de lunes a viernes, a las 7:30 hs hacemos:

```
30 7 * * 1-5 echo "A despertarse!, Hay trabajo por hacer!"
```

La forma más sencilla de ingresar o editar un trabajo con cron, es utilizando el comando $ crontab –e, luego se abrirá el editor de texto predeterminado, con la definición guardada en /var/spool/cron/tabs/.

A partir de esta modalidad con cron, podemos realizar básicamente dos actividades muy importantes:

- Actualizar los datos de nuestra RRD.
- Actualizar la gráfica de datos de la RRD.

Ejemplo de cron para actualizar datos en el archivo RRD:

```
$ sudo cat /etc/crontab
*    *     *  *  *  /home/leopoldo/actualizar_rrd.sh
1,31 10-22 *  *  *  /home/leopoldo/grafica-1.sh >> /dev/null 2>&1
```

El script ‘actualizar_rrd.sh’ actualiza la base de datos con nuevos datos.

El script ‘grafica-1.sh’ genera el gráfico y lo copia a la carpeta ‘/srv/www/htdocs/’ para usarlo en el servicio web.

Protocolo SNMP.

Una forma alternativa de colectar datos de los hosts de cálculo es a través del protocolo SNMP. Este protocolo fue desarrollado con la idea de supervisión y gestión de un grupo de hosts o dispositivos de una red informática. En cada sistema gestionado se ejecuta un componente de software agente que reporta la información al sistema servidor (NMS). Los agentes SNMP exponen los datos de gestión en los sistemas administrados como variables. Estas variables son accesibles a través de comandos SNMP están organizadas en jerarquías. Estas jerarquías, y otros metadatos (tales como el tipo y la descripción de la variable), se describen en las Bases de Información de Gestión (MIB). [14]

Los dispositivos administrados son supervisados y controlados usando cuatro comandos SNMP básicos:

- El comando de lectura es usado por un NMS para supervisar elementos de red. El NMS examina diferentes variables que son mantenidas por los dispositivos administrados.
  snmpget [-p <puerto>] host-community nombre-de-variable [variable]...

- El comando de escritura es usado por un NMS para controlar elementos de red. El NMS cambia los valores de las variables almacenadas dentro de los dispositivos administrados.
  snmpset [-p puerto] [-d] [-r reintentos] [-t timeout] host community nombre-de-variable...

- El comando de notificación es usado por los dispositivos administrados para reportar eventos en forma asíncrona a un NMS. Cuando cierto tipo de evento ocurre, un dispositivo administrado envía una notificación al NMS.

- Las operaciones transversales son usadas por el NMS para determinar qué variables soporta un dispositivo administrado y para recoger secuencialmente información en tablas de variables, como por ejemplo, una tabla de rutas.

Recordamos que Rrdtool, tiene como requerimiento, conocer el momento de cada entrada de datos y también el momento en el que la primera entrada de la base de datos es realizada. Rrdtool no aceptará ningún dato antes de ese momento. Se desea

formular una manera alternativa de recoger datos de los hosts de cálculo, ahora mediante el protocolo SNMP.
Con el siguiente script, podríamos crear una base de datos RRD, y alojar en él datos obtenidos de los agentes SNMP:

```
#!/bin/sh
rrdtool create bandwidth.rrd --start N DS:in:COUNTER:600:U:U \
DS:out:COUNTER:600:U:U RRA:AVERAGE:0.5:1:432
```

El parámetro 'N' le dice a Rrdtool que la hora de inicio es ‘ahora’, ‘in’ es la fuente de datos para registrar la cantidad de tráfico entrante a la red y ‘out’ la cantidad de tráfico saliente a la red.

Un script de ejemplo, podría contener los siguientes comandos para recuperar datos de un nodo de cálculo que corre el agente SNMP, y actualizarlo en la RRD:

```
#!/bin/sh
while true; do
sleep 30
rrdupdate /home/leopoldo/examplesnmp.rrd N:\
`/usr/bin/snmpget -v 1 -c general -Oqv 192.168.0.22 IF-MIB::ifInOctets.2`:\
`/usr/bin/snmpget -v 1 -c general -Oqv 192.168.0.22 IF-MIB::ifOutOctets.2`:\
done
```

El parámetro ‘-v 1’ indica la versión de SNMP a utilizar, ‘-c general’ es la comunidad en la que escucha el agente, la dirección IP del agente, y por último la variable de sistema que deseamos leer. Las variables representan la cantidad de datos de entrada y salida respectivamente de una de sus interfaces de red. Los datos leídos, se almacenan en la base de datos ‘examplesnmp.rrd’.

Es así como, tanto con ‘cron/crontab’ y con ‘snmp’, es posible efectuar la recolección de datos a los efectos de su registración en las bases de datos RRD.

4.4.2. Funciones de consolidación de datos

Las funciones de consolidación de datos tratan sobre unidades de tiempo y datos denominadas Punto de Datos Primarios (PDP) y Consolidado de punto de datos (CDP) [15]

Todo se procesa como una tasa.

Una ‘tasa’ significa un número de ‘algo’ por un intervalo de tiempo. Si por ejemplo, se crea una imagen con la tasa en el eje Y, y la cantidad de tiempo en el eje X, se obtiene el número total de ‘algo’ durante ese intervalo de tiempo. La superficie del rectángulo (x veces y) es la cantidad total. Esto es importante de definir desde un principio, dado que no estamos buscando únicamente el valor Y, sino que podemos estar interesados en el valor de X veces Y.

El hecho de poder registrar la actividad de cada intervalo de un minuto de un total de diez años, es sin duda demasiado trabajo. En primer lugar, el despliegue de esta definición consume demasiados recursos, en segundo lugar es probable que no sea de interés la información con tal nivel de detalle sobre un período tan largo de tiempo. Por ese motivo existe la idea de consolidación, en el caso de RRDtool, se trata de un paquete de un gran número de puntos de datos primarios (PDP's) en un punto de consolidación de datos (CDP).

La siguiente figura (Fig-6) muestra lo que asumimos como un archivo de datos RRA donde, cada CDP es un minuto por valor de datos, y la otra un CDP de tres minutos cada uno.

La superficie de cada uno de estos espacios es igual, lo que resulta en la misma cantidad de, por ejemplo, bytes transferidos en la misma cantidad de tiempo: 0x1 + 3x1 + 0x1 + 1x1 + 2x1 + 3x1 = 9, vs. 1x3 + 2x3 = 9.

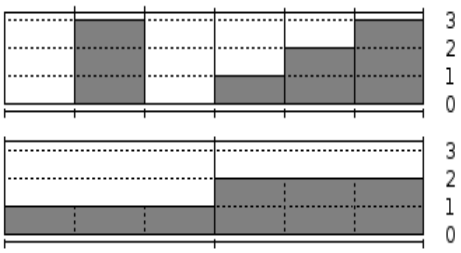


Fig-6

Función de consolidación (CF)

En cualquier caso en que observamos valores expresados en promedios, sean éstas tasas altas o bajas, tienden hacia la tasa promedio. En ocasiones, se hace necesario conocer la tasa más alta o la más baja registrada, para ello esos valores también deben estar guardados, solo basta con crear un archivo RRA en la base de datos RRD, con la función de consolidación deseada.

RRDtool puede recordar, por ejemplo, valores máximos utilizando el RRA con el máximo CF que se ha podido recuperar. Del mismo modo para el valor mínimo, promedio y último.

CF	descripción
MIN	recordar la tasa más baja de todos los puntos de datos PDPs en cada punto de consolidación de datos CDP
AVERAGE	recordar la tasa promedio de todos los puntos de datos PDPs en cada punto de consolidación de datos CDP
MAX	recordar la tasa más alta de todos los puntos de datos PDPs en cada punto de consolidación de datos CDP
LAST	recordar el último de todos los puntos de datos PDPs en cada punto de consolidación de datos CDP

Tomando como ejemplo los valores de la Fig-4, tenemos:

Función de Consolidación	Valores -> Resultado
MIN	(0,3,0),(1,2,3) -> (0),(1)
AVERAGE	(0,3,0),(1,2,3) -> (1),(2)
MAX	(0,3,0),(1,2,3) -> (3),(3)
LAST	(0,3,0),(1,2,3) -> (0),(3)

En este sentido, la herramienta RRDtool, incorpora un comando: GPRINT que permite imprimir propiedades de sus gráficos, como por ejemplo, el valor mínimo, medio y máximo. La siguiente tabla muestra lo que resultaría al utilizar datos de la Fig-4:

GPRINT	Seis intervalos cortos	Dos intervalos largos
Minimum	(0,3,0,1,2,3)->0	(1,2)->1
Average	(0,3,0,1,2,3)->1.5	(1,2)->1.5
Maximum	(0,3,0,1,2,3)->3	(1,2)->2

Si ahora volvemos a usar la función GPRINT para la impresión del máximo de todos los máximos, va a imprimir tres en vez de uno. Esto es lo que se desea y espera. Del mismo modo se obtiene cero en lugar de uno cuando GPRINT imprime el mínimo de todos los mínimos.

Declaraciones CDEF.

La definición de las funciones de consolidación (CF's), se establecen en la creación de la base de datos RRD. El script mencionado anteriormente, y que genera nuestro primer ejemplo en el uso de la herramienta RRDtool, definía:

```
leopoldo@linux-jos:~> cat crear_rdd.sh

#!/bin/sh

 rrdtool create test.rrd             \
   --start 1493992800  --step 300      \
            DS:speed:COUNTER:600:U:U   \
            RRA:AVERAGE:0.5:1:24       \
            RRA:AVERAGE:0.5:6:10
```

'test.rrd': nombre del archivo de base de datos.
'speed': nombre de la variable que aloja el dato, de tipo contador.

Ahora bien, el script que mencionamos en la generación del gráfico de la Fig-2, puede ser tomado como ejemplo:

```
leopoldo@linux-jos:> cat grafico2.sh
#!/bin/sh
   rrdtool graph speed2.png                          \
      --start 1493993400 --end 1493998000            \
      --vertical-label m/s                           \
      DEF:myspeed=test.rrd:speed:AVERAGE             \
      CDEF:realspeed=myspeed,1000,\*                 \
      LINE2:realspeed#FF0000
```

En este script, es utilizada la definición 'speed' del archivo test.rrd, en la definición de una función de consolidación 'myspeed' de tipo 'Average' o promedio, mediante la

declaración ‘DEF’. La función ‘CDEF’ es utilizada en un cálculo, que define a ‘realspeed’ a partir del valor de ‘myspeed’ multiplicado por un mil (1000).

Las declaraciones ‘CDEF’ son utilizadas en la realización de cálculos sobre variables definidas en los archivos RRD. [16]

La sintaxis es:

```
DEF:var_name_1=some.rrd:ds_name:CF
CDEF:var_name_2=RPN_expression
```

Aquí, es definida ‘var_name_1’ para alojar datos de la fuente de datos ‘ds_name’ que se encuentra en el archivo RRD ‘some.rrd’ con la función de consolidación ‘CF’.

Expresiones RPN.

RPN es la abreviatura de notación polaca inversa o ‘Reverse Polish Notation’. Funciona de la siguiente manera, se definen las variables o números en una pila, y las operaciones, para luego procesarlas. El resultado es alojado en la pila.

En un caso de multiplicación por ocho (8) se verá así:

1. Comenzar con una pila vacía
2. Colocar el contenido de la variable ‘inbytes’ en la pila
3. Colocar el número ocho en la pila
4. Colocar la operación multiplicación (*) en la pila
5. Procesar la pila
6. Recuperar el valor de la pila y colocarlo en la variable ‘inbits’

Ahora bien, vamos a hacer un ejemplo con números reales. Supongamos que la variable ‘inbytes’ posee el valor 10, el resultado de la pila sería:

1. ||
2. | 10 |
3. | 10 | 8 |
4. | 10 | 8 | * |
5. | 80 |
6. ||

Para otras operaciones como la resta y la división el orden de los factores si es importante, no así para la multiplicación. En términos generales tiene el siguiente orden:

```
y = A - B  -->  y=minus(A,B)  -->  CDEF:y=A,B,-
```

Para desarrollar un despliegue correcto en la definición de estos cálculos, en primer lugar, se debe obtener una imagen clara de lo que se quiere hacer. Descomponer el problema en partes más pequeñas hasta que no se pueden dividir más. Entonces es bastante simple para convertir las ideas en expresiones RPN.

Como ejemplo, suponemos disponer del registro de contadores para cada enlace WAN de un router que se está supervisando:

```
router1.rrd, 'link1in', 'link2in'
router2.rrd, 'link1in', 'link2in'
router3.rrd, 'link1in', 'link2in'
```

Suponemos además, que deseamos sumar todos estos contadores, excepto el contador 'link2in' de router2.rrd. En este ejemplo, "router1.rrd: link1in" refiere a la fuente de datos (DS) 'link1in' del archivo 'router1.rrd', lo que nos queda:

```
router1.rrd:link1in
router1.rrd:link2in
router2.rrd:link1in
router3.rrd:link1in
router3.rrd:link2in
-------------------   +
Resultado de la suma
```

Como función matemática a aplicar, podemos definir:

```
add(router1.rrd:link1in,router1.rrd:link2in,router2.rrd:link1in,
router3.rrd:link1in,router3.rrd:link2.in)
```

La aplicación a partir de la herramienta RRDtool, definimos:

```
DEF:a=router1.rrd:link1in:AVERAGE
DEF:b=router1.rrd:link2in:AVERAGE
DEF:c=router2.rrd:link1in:AVERAGE
DEF:d=router3.rrd:link1in:AVERAGE
DEF:e=router3.rrd:link2in:AVERAGE
```

Ahora, la función matemática se convierte en: add(a, b, c, d, e)

En expresiones RPN, no hay ningún operador que sume más de dos valores por lo que hay que hacer varias adiciones. Se agrega a y b, c para añadir el resultado, luego añadir d para el resultado y añadir e para el resultado. Una forma de reducir la expresión puede ser: ((((a+b) + c) + d) + e) >.

En notación RPN, la expresión se reduce a:

```
CDEF:result=a,b,+,c,+,d,+,e,+
```

El proceso de esta expresión, en el paso a paso, sería:

```
push value of variable a on the stack: a
   push value of variable b on the stack: a b
   push value of variable c on the stack: a b c
   push value of variable d on the stack: a b c d
   push value of variable e on the stack: a b c d e
   push operator + on the stack:          a b c d e +
   and process it:                        a b c P   (where P == d+e)
   push operator + on the stack:          a b c P +
   and process it:                        a b Q     (where Q == c+P)
   push operator + on the stack:          a b Q +
   and process it:                        a R       (where R == b+Q)
   push operator + on the stack:          a R +
   and process it:                        S         (where S == a+R)
```

Como se puede observar, la expresión RPN 'a, b, c, d, e, +, +, +, +, +' evaluará '((((d + e) + c) + b) + a)', inicia la operación de derecha hacia izquierda. Tendría el mismo resultado que 'a, b, +, c, +, d, +, e, +'. Esto se conoce como la ley conmutativa de la suma.

En caso de que una expresión contiene una multiplicación, 'result = a + b * c', se comienza con la multiplicación de 'b * c' y luego se suma 'a' al resultado. Es posible alterar la posición de B y C, no debe alterar la posición de a y b.

```
CDEF:result= a,b,c,*,+
```

Esta última expresión podría ser escrita de la siguiente manera:

```
CDEF:result= b,c,*,a,+
```

Valores desconocidos.

En determinadas circunstancias, la recolección de datos falla. Esto puede ser común, especialmente cuando, por ejemplo, se consultan sus estados a través de enlaces de datos ocupados. RRDtool puede ser configurado para permitir un valor desconocido o incluso más y calcular la actualización que falta. Por ejemplo, para consultar un dispositivo cada minuto, se crea un llamado PDP o punto de datos primarios por minuto. Si también hemos definido el RRD para contener un DRR que almacene los

valores de 5 minutos, se necesitan leer cinco de esos PDP para crear una CDP (punto de datos consolidada).
Los PDPs pueden llegar a ser desconocido en dos casos:

- Las actualizaciones se registran con demasiada separación. Esto se mejora mediante el ajuste de "latido".
- La actualización se establece en desconocido a propósito mediante la inserción de ningún valor (utilizando la opción de plantilla) o mediante el uso de "U" como valor ingresado.

Cuando se calcula un CDP, otro mecanismo determina si el CDP es válido o no. Si hay demasiados PDP desconocidos, el CDP se desconoce también. Está determinado por el factor 'xff' (x files factor). Si sólo se permite un PDP desconocido por CDP, esto hará que el CDP sea desconocido.

Supongamos, con un ejemplo, tener un contador que incrementa en uno por segundo y es recuperado cada minuto:

```
Valor del contador   tasa        resultado
   10'000
   10'060            1;          (10'060-10'000)/60 == 1
   10'120            1;          (10'120-10'060)/60 == 1
   unknown          ;    no sabe el último valor
   10'240            unknown;    no sabe el valor previo
   10'300            1;          (10'300-10'240)/60 == 1
```

En este caso, si el CDP debía ser calculado a partir de las últimas cinco actualizaciones, se consiguen dos PDP 'desconocidos' y tres PDP conocidas. Si el factor xff habría sido ajustado a 0,5, que por cierto es un factor de utilización común, el CDP tendría un valor conocido de 1. Si xff habría sido ajustado a 0,2 el CDP resultante sería 'desconocido'.

Se hace necesario decidir valores adecuados para el 'heartbeat', el número de PDP por CDP y el factor xff; dado que definen el comportamiento de la RRA.

Trabajar con datos desconocidos en la base de datos.

Falsas lecturas determina el ingreso de valores 'desconocidos' en un archivo RRA. Si a posterior se hacen cálculos con este tipo de valor, el resultado tiene que ser 'desconocido' también. Esto significa que una expresión como resultado = a, b, + será desconocida si A o B es desconocido. Sería un error pasar por alto el valor desconocido y devolver el valor del otro parámetro. Al hacerlo, podría asumir "desconocido" como "cero" y eso no es cierto.

Un caso de ejemplo, que es común en el ambiente, es que alguien recoge datos durante, por ejemplo, más de un año. Luego, un nuevo elemento se ha instalado en el equipo que es monitoreado, un nuevo RRD es creado y los 'scripts' se cambian para agregar un contador de la antigua base de datos y un contador de la nueva base de

datos. El resultado decepciona, dado que una gran parte de las estadísticas desaparecen misteriosamente. Esto ocurre porque los valores de la antigua base de datos (valores conocidos) se añadieron a los valores de la nueva base de datos (valores desconocidos) y el resultado por lo tanto, es 'desconocido'.

Infinitos.

Datos infinitos es otra forma de un número especial. No es posible graficarlos porque, por definición, nunca se alcanzará el valor infinito. Si es posible pensar en el infinito positivo y negativo dependiendo de la posición relativa de cero.

RRDtool es capaz de representar el infinito al detener en su corriente máxima (para el infinito positivo) o mínimo (para el infinito negativo) sin conocer este máximo (mínimo).

Infinito en RRDtool se utiliza sobre todo para dibujar un área sin conocer sus dimensiones verticales. Es posible asimilar esta situación, en dibujar una zona con una altura infinita y mostrar sólo la parte que es visible en el gráfico actual. Esta es probablemente una buena manera de aproximar el infinito y que permite algunos formalizar algunos trucos.

Puede que en ciertas circunstancias, se desea descartar datos 'desconocidos' y pretender con ello que sean cero (o cualquier otro valor para el caso) y en otras, pretender que los datos conocidos sean 'desconocidos' (para descartar los datos conocidos-malos). Esta es la razón por la cual las definiciones CDEF tienen soporte para datos desconocidos.

4.4.3. Tipos de fuentes de datos (Data Sources)

La estructura de una base de datos RRD es diferente a otras bases de datos lineales, que definen tablas y columnas, y muchos otros parámetros más. Dependiendo el caso, estas definiciones son consideradas complejas, especialmente en grandes bases de datos. [17]

Las bases de datos RRDtool, se utilizan principalmente para fines de monitoreo y vigilancia, y por lo tanto son muy simples en su estructura. Los parámetros que la definen son variables que tienen valores y archivos de esos valores.

Debido a su estructura, la definición de una base de datos RRDtool también incluye una disposición para especificar acciones específicas a tomar en la ausencia de valores de actualización.

Las siguientes terminologías serán utilizadas en las RRD: Fuente de datos (DS), hearbeat, Data Source Type (DST), archivos Round Robin (RRA), y funciones de consolidación (CF).

En la creación de una base de datos RRD se definen básicamente dos aspectos: las fuentes de datos DS, y las funciones de consolidación (CF) de esos datos. La forma tiene el siguiente aspecto:

```
$ rrdtool create filename [--start time] [--steps] \
[ DS:ds-name:DST:heartbeat:min:max]                \
[ RRA:CF:xff:step:rows]
```

Las fuentes de datos DS, se especifican de la siguiente manera:

```
DS:ds-name:DST:heartbeat:min:max
```

Una base de datos RRD puede aceptar la entrada de varias fuentes de datos (DS), y definir las propiedades básicas de cada una para su almacenamiento. Definiciones:

En la definición, *ds-name* es el nombre que utilizará para definir el valor de la fuente de datos de una RRD, su nombre debe contener de 1 a 19 caracteres de longitud utilizando [a-z A-Z 0-9_].

DST define el tipo de origen de datos. El resto de los argumentos de una entrada de fuente de datos dependen del tipo de fuente de datos. Opciones:

- *GAUGE* almacena cantidades como ser la temperatura, la cantidad de memoria RAM usada por el sistema operativo.
- *COUNTER* cuenta, y aumenta el valor de forma continua, jamás disminuye, salvo desbordamientos.
- *DERIVE* almacena una tasa entre el último valor y el valor actual. Su valor puede disminuir. No hay control de desbordamiento.
- *ABSOLUTE* almacena valores que se restablecen después de cada lectura. Es útil para contadores rápidos que tienden a desbordarse.

heartbeat define el tiempo entre cada actualización de valor en la base de datos. Si se excede este tiempo, el valor se almacenará como valor “NaN”.

Junto a las fuentes de datos DS, y seguido a ellas, se define funciones de consolidación de archivo, que consolidan los puntos de datos primarios PDP a través de una función de agregación. Las funciones aceptadas son: AVERAGE, MIN, MAX, LAST.

El formato de una línea de definición RRA es:

```
RRA: AVERAGE|MIN|MAX|LAST :xff:steps:rows
```

Donde:

xff define qué parte de un intervalo de consolidación puede estar compuesta de ‘datos desconocidos’ mientras que el valor consolidado sigue siendo considerado como conocido. Se da como la relación de PDP permitidos ‘DESCONOCIDO’ con el número de PDP en el intervalo, varía de 0 a 1.

steps: define la cantidad de muestras de un archivo para consolidar. Para el cálculo de la resolución, se tiene en cuenta el parámetro step.

rows: define cómo se mantienen o conservan las generaciones de valores de datos en una RRA, como la cantidad de registros o filas a guardar.

Como ejemplo, si deseamos registrar un valor durante un período dado en segundos, con un valor de actualización de la RRD, de por ejemplo -*–step=300*, medido en segundos, ¿cuántos registros o *rows* necesito para su almacenamiento?, la respuesta es:

86400 seg [1 día] / 300 segundos [5 minutos] = **288**
604800 seg [7 días] / 300 segundos [5 minutos = **2016**
604800 seg [7 días] / 900 segundos [15 minutos] = **672**
2678400 seg [31 días] / 900 segundos [5 minutos] = **2976**
31536000 seg [365 días] / 86400 segundos [1 día] = **365**

A medida que aumenta la resolución de datos requerida, aumenta la cantidad de filas necesarias para su almacenamiento. Se hace necesario definir los valores de máxima, mínima y promedio que se desean obtener, y el muestreo de datos necesario para su medición.

Valor de step en la RRA	**Frecuencia con que se guarda el valor**	**Número obtenido**
1	Cada vez	1 * 300s = 5m
3	Cada tercera vez	3 * 300s = 15m
6	Cada sexta vez	6 * 300s = 30m
12	Cada 12va vez	12 * 300s = 1h
72	Cada 74va vez	72 * 300s = 6h
144	Cada 144va vez	144 * 300s = 12h
288	Cada 288va vez	288 * 300s = 24h

En otras palabras y a manera de resumen, para obtener y tomando el valor de step en 300 segundos, y:

Si quiero obtener el promedio de :	**… defino:**
1 día con resolución de 5 minutos	RRA:AVERAGE:0.5:1:288
1 semana con resolución de 15 minutos (604800 / 900 [15 minutos]) = 672	RRA:AVERAGE:0.5:3:672
1 mes con resolución de 1 horas (2678400 [31 días]/3600 [1 hora]) = 744	RRA:AVERAGE:0.5:12:744
1 año con resolución de 6 horas (31536000 [365 días]/21600 [6 horas]) = 1460	RRA:AVERAGE:0.5:72:1460

De la misma manera se definen estos parámetros para valores MIN, MAX y LAST.

4.4.4. El heartbeat y step.

RRDtool recibe muestras de datos, de manera controlada y planificada, y en ocasiones arbitrarias, acumuladas en los puntos de dato primario PDP de cada intervalo *step*. [17]

El valor de *heartbeat* define el intervalo máximo aceptable entre las muestras. Si el intervalo entre muestras es inferior al heartbeat, entonces una tasa promedio se calcula y se aplica para ese intervalo. Si el intervalo entre muestras es más largo, entonces todo ese intervalo se considera "desconocido" y se lo almacena como tal. Hay otras situaciones que pueden hacer que un intervalo de muestreo sea "desconocido", como la tasa que excede los límites, o una muestra que estuvo marcada explícitamente como desconocido.

El heartbeat puede tener un valor corto (poco común) o largo (típico) con respecto al intervalo de "paso" entre los PDP. Un heartbeat corto significa que se requieren múltiples muestras por un PDP, y si no los consigue marcará el PDP como 'unknow'. Un heartbeat largo puede abarcar varios "pasos" o steps, lo que significa que es aceptable tener múltiples PDP calculados a partir de una sola muestra.
Un ejemplo extremo de esto podría ser un step de 5 minutos y un heartbeat de un día, en cuyo caso, una sola muestra de todos los días se traducirá en todas PDP's para todo ese período de días.

4.4.5. Muestreo de Datos

Una característica importante y singular de RRDtool es la técnica utilizada para el muestreo de datos, dado que es prácticamente imposible recoger datos y guardarlos en intervalos exactos en extremo. La técnica de RRDtool es interpolar datos, los que se almacenan en intervalos exactos.

Supongamos que un contador se incrementa en uno exactamente cada segundo, y se desea medirlo en intervalos de 300 segundos. Entonces, se deben recuperar valores en exactamente 300 partes. Sin embargo, y debido a diversas circunstancias habrá segundos de retraso y el intervalo será por ejemplo de 303. El delta también será 303 en ese caso. Obviamente, RRDtool no debe poner 303 en la base de datos y hacer creer que el contador se incrementó en 303 en 300 segundos. Para estos casos RRDtool interpola datos: es decir, altera el valor 303 como si hubiera sido almacenado anteriormente y será 300 en 300 segundos. Lo mismo será para el caso en que sean 297 segundos y también el contador incrementado en 297. De nuevo, RRDtool interpola y almacena 300 como fue establecido.

Un ejemplo para graficar el concepto:

en la RRD	**en la realidad**
time+000: 0 delta="U"	time+000: 0 delta="U"
time+300: 300 delta=300	time+300: 300 delta=300
time+600: 600 delta=300	time+603: 603 delta=**303**
time+900: 900 delta=300	time+900: 900 delta=**297**
U=Unknow	

Otro Ejemplo: Vamos a crear dos bases de datos idénticas, el objeto es registrar un contador y un valor promedio:

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

Ahora hacemos una copia de la base para obtener la segunda RRD:

```
$ cp seconds1.rrd seconds2.rrd
```

Se ingresan datos, apenas diferentes en cada archivo:

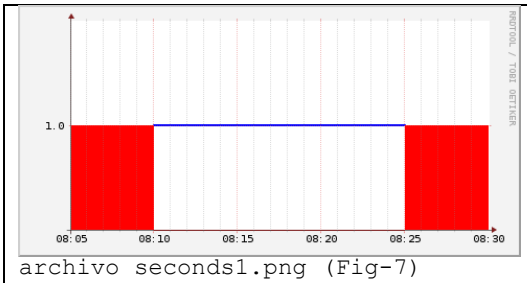
```
 $ rrdtool update seconds1.rrd \
      920805000:000 920805300:300 920805600:600 920805900:900
 $ rrdtool update seconds2.rrd \
      920805000:000 920805300:300 920805603:603 920805900:900
```

Generamos un gráfico –seconds1.png y seconds2.png- a partir de los datos de cada base de datos:

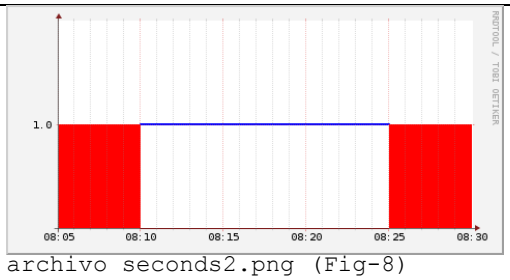
```
$ rrdtool graph seconds1.png                        \
      --start 920804700 --end 920806200              \
      --height 200                                   \
      --upper-limit 1.05 --lower-limit 0.95 --rigid \
      DEF:seconds=seconds1.rrd:seconds:AVERAGE       \
      CDEF:unknown=seconds,UN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

$ rrdtool graph seconds2.png                        \
      --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
```

Al visualizar ambos gráficos en el navegador web se obtiene:



archivo seconds1.png (Fig-7)



archivo seconds2.png (Fig-8)

Resultado: Ambos gráficos (Fig-7 y Fig-8) muestran la misma salida, a pesar de que la entrada de datos es apenas diferente. Esto se debe por la interpolación de datos que efectúa RRDtool sobre el momento en que se registran datos en los archivos RRD.

4.5. Comparativa con otros sistemas de Base de Datos, el caso MySQL

Con el objeto de poder comprender mejor el funcionamiento de la herramienta RRDtool, de su técnica para bases de datos RRD de series de tiempo, se presenta un ejemplo tomando como contra parte al producto MySQL. Deseamos reproducir el funcionamiento de los archivos RRD's, y observar los tiempos de respuestas y eficiencia obtenido con MySQL. Esta actividad, se basa en un documento en el que se describen los pasos, 'Round-Robin Database Storage Engine (RRD)' de Oli Sennhauser (© GNU FDL). [18]

Para la implementación del ejemplo: Se asume que la tabla 'statistic', que creamos en la Base de Datos 'rrd_test', de MySQL, es la que deseamos convertir a RRD.

```
CREATE TABLE statistic (
    attribute_key     INT UNSIGNED    NOT NULL DEFAULT '0'
  , start_utime       INT UNSIGNED    NOT NULL DEFAULT '0'
  , end_utime         INT UNSIGNED             DEFAULT NULL
  , logging_interval  INT UNSIGNED    NOT NULL DEFAULT '0'
  , value             BIGINT UNSIGNED NOT NULL DEFAULT '0'
  , PRIMARY KEY (attribute_key, start_utime)
  , KEY start_time (start_utime)
)
;
```

Luego se añade una tabla 'statistic_rrd', necesaria para simular el comportamiento RRD. Consideramos el formato de fila FIJA de MySQL para aumentar la velocidad y permitir inserciones concurrentes.

Supongamos que queremos almacenar 25920 filas, unos 11 Mbyte de datos, entonces, la tabla debería incluir la siguiente configuración:

```
CREATE TABLE statistic_rrd (
    rrd_key           INT UNSIGNED    NOT NULL AUTO_INCREMENT
PRIMARY KEY
  , attribute_key     INT UNSIGNED    NOT NULL DEFAULT '0'
```

```
    , start_utime       INT UNSIGNED    NOT NULL DEFAULT '0'
    , end_utime         INT UNSIGNED             DEFAULT NULL
    , logging_interval  INT UNSIGNED    NOT NULL DEFAULT '0'
    , value             BIGINT UNSIGNED NOT NULL DEFAULT '0'
    , UNIQUE KEY (attribute_key, start_utime)
    , KEY start_time (start_utime)
  )
  ROW_FORMAT = FIXED, MAX_ROWS = 25920 ;
```

Queda por agregar una tabla donde almacenar la rrd_key e inicializar la clave:

```
CREATE TABLE statistic_rrd_key (
    rrd_key BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY
)
;
INSERT INTO statistic_rrd_key VALUES (0);
```

La Lógica RRD.

Para simular el comportamiento RRD se necesita agregar un disparador en cada INSERT, y fijar el número de filas. Dado que RRDtool reemplaza datos en la RRD, en nuestro ejemplo, los valores más antiguos serán 'reemplazados' por nuevos valores, por ello se requiere en este caso, el cambiar las sentencias INSERT por REPLACE y adaptar las sentencias UPDATE y DELETE a la nueva estructura de la tabla. La instrucción tendrá el siguiente aspecto:

```
 REPLACE INTO statistic_rrd
   (attribute_key, start_utime, end_utime, logging_interval, value)
  VALUES
   (ROUND(RAND()*100), UNIX_TIMESTAMP(NOW()), NULL, 100, 123456789)
 ;
 SELECT * FROM statistic_rrd;
 SELECT * FROM statistic_rrd_key;
```

Se ingresan datos a la tabla, con una instrucción como:

```
REPLACE INTO statistic_rrd
    (attribute_key, start_utime, end_utime, logging_interval,
value)
 VALUES
    (ROUND(RAND()*100000), UNIX_TIMESTAMP(NOW()), NULL, 100,
123456789)
;
```

Decimos que si necesitamos almacenar un valor, cada 5 minutos durante 90 días, esto es ((60seg*24horas/5 minutos)*90días), será un total de 25920 elementos. Definimos entonces ese valor como la máxima cantidad de filas a ingresar.

Para la inserción de datos utilizo un script PHP que contiene básicamente una instrucción de conexión a la BD Mysql, y luego realiza la inserción de datos mediante REPLACE.

Ahora bien, nuestra prueba, se realiza sobre una máquina virtual soportada sobre Vmware Workstation 9.0, con una asignación de 1GB de RAM, 1 core CPU Pentium(R) Dual-Core CPU T4400 a 2.20GHz y disco virtual de 15GB. Los resultados son los siguientes:

```
leopoldo@miLinux:> php -a -e -f inserta.php

Interactive mode enabled

PDO connection object created
773.63790714716 inserciones /seg
cantidad de filas: 25920
tiempo utilizado: 33.504045963287
```

```
Al realizar un select * de la tabla en la base de datos, arroja: 25913
rows
```

Se observa que no se han insertado los 25920 elementos, y surge la pregunta ¿qué paso?, la respuesta es: sencillamente no lo hizo. Ahora bien, tomando la decisión de aumentar el número de inserciones, por ejemplo 25920*5=129600, modificamos el script, y el resultado arroja:

```
leopoldo@miLinux:~/especializa/mysql> php -a -e -f inserta.php
Interactive mode enabled

PDO connection object created
1007.9561338823 inserciones /seg
cantidad de filas: 129600
tiempo utilizado: 128.577023983
```

```
Al realizar un select * de la tabla en la base de datos, arroja:
129533 rows
```

Una vez más se observa que tampoco se ha llegado al valor de 129600 elementos, aunque si se hayan logrado mayor cantidad de inserciones por segundo.

La conclusión que se obtiene está en directa relación con el objetivo previsto, dado que MySQL como herramienta, y por distintos motivos, no logra realizar la inserción del total de registros requeridos. Es así que RRDtool, aprovecha esta situación para interpolar datos, y manejarlos de una manera coherente en el tiempo. Este detalle toma particular importancia al momento del análisis de los datos en función de un momento dado.

4.6. Instalación de la herramienta RRDtool.

Básicamente hay dos formas de realizar la instalación de la herramienta. Una de ellas es descargar las fuentes del sitio en línea para luego realizar la compilación y puesta a punto. La forma de instalación alternativa se corresponde a la manera tradicional de instalar una aplicación bien conocida, en función a la distribución de Linux y Windows que se tenga como sistema operativo de base. [19]

Para sistemas Linux, en el caso de OpenSUSE / SUSE la instalación consiste en:

```
# zypper addrepo http://download.opensuse.org/repositories/devel:
languages:python/openSUSE_13.2/devel:languages:python.repo
# zypper refresh
# zypper install rrdtool
# which rrdtool
/usr/bin/rrdtool
```

Luego de la instalación de la herramienta, se pueden observar los archivos creados en la carpeta de destino de las RRD's, ver tabla (TAB-1).

Para sistemas Linux distribución Ubuntu, la instalación simplificada consiste en:

```
# apt-get install libpango1.0-dev libxml2-dev
# apt-get install librrds-perl rrdtool
```

En el caso en que la modalidad de instalación que se prefiera, sea la de bajar los archivos fuentes, el procedimiento general que se puede definir es el siguiente:

```
# wget http://oss.oetiker.ch/rrdtool/pub/rrdtool-1.4.7.tar.gz
# tar -zxvf rrdtool-1.4.7.tar.gz
# cd rrdtool-1.4.7
# mkdir /tmp/rrdbuild
# export BUILD_DIR=/tmp/rrdbuild
# mkdir /opt/rrdtool-1.4.7
# export INSTALL_DIR=/opt/rrdtool-1.4.7
# ./configure -prefix=$INSTALL_DIR
# make
# make install
```

En el proceso de instalación por compilación de fuentes, se tiene acceso a variadas banderas (flags) para realizar una parametrización más precisa, que puede ser consultada en el archivo README de la carpeta inicial.

Es posible encontrar mayor precisión sobre instalaciones para sistemas operativos más específicos, en el sitio en línea mencionado.

4.7. Funcionalidades de la herramienta RRDtool.

Tal y como aparece en el sitio web en línea [3], es posible preguntarse: ¿Qué es lo que hace especial a RRDtool como herramienta? Algunas respuestas pueden ser:

- En el caso de las bases de datos lineales –tradicionales-, los nuevos datos se adjuntan siempre en la parte inferior de la tabla de base de datos, por lo tanto su tamaño sigue aumentando indefinidamente. Para una base de datos RRDtool, el tamaño se determina en tiempo de creación y se mantiene en el tiempo. Se debe imaginar una base de datos RRDtool como el perímetro de un círculo. Los datos se agregan a lo largo del perímetro. Cuando nuevos datos llega al punto de

partida, sobrescribe los datos existentes. De esta manera, el tamaño de una base de datos RRDtool siempre permanece constante. El nombre de "Round Robin" se deriva de este comportamiento.

- Otras bases de datos almacenan los valores tal como han sido suministrados. RRDtool, en cambio, se puede configurar para calcular la velocidad de cambio de la anterior para el valor actual y almacenar esta información en su lugar.

- Otras bases de datos se actualizan cuando se le suministran valores. La base de datos RRDtool está estructurada de tal manera que necesita datos a intervalos de tiempo predefinidos. Si no obtiene un nuevo valor en ese intervalo, almacena un valor 'Unknow' para ese intervalo. Por lo tanto, cuando se utiliza la base de datos RRDtool, es imperativo el uso de técnicas, como secuencias de comandos (scripts), que se ejecuten a intervalos regulares para asegurar un flujo constante de datos para actualizar la base de datos RRDtool.
- RRDtool está diseñado para almacenar series temporales de datos. Con cada actualización de datos, un sello de tiempo asociado también es almacenado. El tiempo es siempre expresado en segundos transcurridos desde '01-01-1970'.

- RRDtool posee un conjunto de comandos para llevar a cabo diversas operaciones en las bases de datos RRD. Los comandos se pueden disparar desde un Shell o un script de Perl. Los scripts actúan como contenedores para acceder a datos almacenados en bases de datos RRDtool.

Funciones incluidas en la herramienta RRDtool:

create
Genera una nueva base de datos de Round Robin (RRD).

update
Guarda nuevos valores de datos en una RRD.

graph
Crea un gráfico a partir de datos almacenados en uno o varios RRD. Además de los gráficos que generan, también se pueden extraer los datos a salida estándar en diversos formatos.

graphv
Crea un gráfico a partir de datos almacenados en uno o varios RRD. Igual que graph, pero los metadatos se imprimen antes de la gráfica. Comprobarlo con rrdgraph.

dump
Vuelca el contenido de una RRD en formato ASCII. Es muy útil para mover una RRD de una arquitectura de computadora a otra.

restore

Restaura una RRD en formato XML a una RRD binario.

fetch
Obtiene datos durante un determinado período de tiempo a partir de un RRD. La función graph utiliza fetch para recuperar datos de una RRD. Compruebe rrdfetch.

tune
Modifica la configuración y estructura de un RRD.

first
Devuelve el tiempo o momento, de la primera actualización de un RRD.

last
Devuelve el tiempo o momento, de la última actualización de un RRD.

lastupdate
Encuentra el tiempo de la última actualización de un RRD, y devuelve el valor almacenado para cada punto de referencia en la actualización más reciente.

info
Obtiene información acerca de una RRD.

resize
Cambia el tamaño de las RRA individuales.

xport
Exportar datos recuperados de una o varias RRD.

flushcached
Limpia los valores de un archivo RRD específico en memoria.

En TAB-3 se podrá apreciar un ejemplo que tiene diversas formas de aplicación de esta herramienta.

Detección de Comportamiento Anormal

RRDtool proporciona elementos para la detección en tiempo real de comportamiento anormal de una base de datos. Estos componentes incluyen:

- Un algoritmo para predecir el valor de una serie de tiempo de un intervalo de tiempo en el futuro.
- Una medida de la desviación entre los valores predichos y observados.
- Un mecanismo para decidir si y cuando un valor observado o secuencia de valores observados es demasiado desviada desde el valor previsto.

Estos componentes en detalle:

- El algoritmo de predicción de series de tiempo de Holt-Winters es un algoritmo que predice futuras observaciones en una serie de tiempo de forma adaptativa. Su previsión es la suma de tres componentes: una línea de base (o de intercepción), una tendencia lineal en el tiempo (o pendiente), y un coeficiente estacional (un efecto periódico, tales como un ciclo diario). Hay un coeficiente estacional para cada punto en el período (ciclo) de tiempo. Después de que se observa un valor, cada uno de estos componentes se actualiza. Esto significa que el algoritmo "aprende" de los valores pasados y los utiliza para predecir el futuro. El ritmo de adaptación se rige por 3 parámetros, alfa (intercepción), beta (pendiente) y gamma (estacional). [20]

- La medida de la desviación es una desviación absoluta ponderada estacional. La duración de la desviación se mide por separado para cada punto del ciclo. Al igual que con Holt-Winters, la desviación prevé el uso de la medida calculada a partir de los valores anteriores (pero sólo en ese punto en el ciclo estacional). Después de que se observó el valor, el algoritmo aprende del valor observado a través de suavizado (smoothing) exponencial.

- El comportamiento anormal (fallo potencial) se informa cada vez que el número de veces que el valor observado cumple o excede un umbral fijado dentro de una ventana temporal especificado (por ejemplo 5 violaciones durante los últimos 45 minutos con un valor observado de cada 5 minutos).
- Esta funcionalidad está integrada en un conjunto relacionado de archivos de datos RRA. En particular, un set de RRA registra fallas potenciales. Con estos datos se podría, por ejemplo, utilizar una aplicación front-end para RRDtool para iniciar alertas en tiempo real.

Modalidad de control remoto.

Al iniciar RRDtool con la opción de línea de comandos '-', espera un input a través de la entrada estándar (stdin). Con esta característica se puede mejorar el rendimiento uniendo RRDtool a otro proceso a través de pipas (pipes). A lo largo de estas pipas, RRDtool acepta los mismos argumentos en la línea de comandos y algunos comandos especiales como *cd*, *mkdir*, pwd, *ls* y *quit*.

Cuando la ejecución de un comando finaliza, RRDtool imprimirá la cadena 'OK', seguido de la información de temporización de la forma u: UserTime s: SYSTEMTIME. Ambos valores son los totales acumulados de segundos desde que RRDtool se inició. Si se produce un error, una línea con el texto 'ERROR: Descripción del error' será impreso.

En la mayoría de los casos, RRDtool no abortará, a menos que algo realmente serio suceda. Si se especifica un directorio de trabajo (workdir) y el UID es 0, RRDtool hará un ‘chroot’ a ese directorio. Si el UID no es 0, RRDtool sólo cambia el directorio actual al directorio de trabajo.

El modo RRD server.

Es posible trabajar en la modalidad RRD-Server para ser accedido remotamente mediante sockets. Para ello se debe elegir un puerto TCP/IP y añadirlo a /etc/services como por ejemplo:

```
rrdsrv      13900/tcp                       # RRD server
```

Es posible utilizar cualquier número de puerto no utilizado en el archivo de servicios, siempre que el servidor y el sistema cliente utilicen la misma configuración de puerto.

Es posible agregar a RRDtool como meta-servidor al archivo de configuración /etc/inetd.conf:

```
rrdsrv stream tcp nowait root /opt/rrd/bin/rrdtool rrdtool - /var/rrd
```

Se necesita crear el directorio de base de datos, que por default es '/var/rrd' y reinicializar el sistema inetd. Si todo fue configurado correctamente, será posible acceder al servidor con por ejemplo, sockets de Perl, herramientas como 'netcat', o en un test interactivo mediante el uso de ' telnet localhost rrdsrv'.

RRDCACHED, el demonio CACHING

Para configuraciones muy grandes, la actualización de miles de archivos RRD a menudo se convierte en un grave problema de IO. Para estos casos, es posible utilizar rrdcached, un demonio de almacenamiento en caché para RRDtool que puede ayudar a disminuir la presión sobre el uso de los discos.

4.8. RRDtool y Ganglia

4.8.1. Presentación de Ganglia.

GANGLIA es un sistema informático basado en web open Source, para el seguimiento distribuido escalable de sistemas de computación de alto rendimiento, como clústeres y Grids. Se basa en un diseño jerárquico dirigido a monitorear agrupamientos de clústeres. [21]

La especial característica que posee esta herramienta, es la facilidad de su instalación por estar soportada en web. Integra un modo 'móvil' para ser utilizado en tabletas y teléfonos, con fines de monitoreo remoto.

Sorprende de inmediato la calidad de los gráficos utilizados para mostrar información de uso y estado en línea de recursos computacionales como ser: CPU, Memoria, Red. Para la visualización, es posible definir estancias de tiempo como un día, una semana, un mes, un año.

La herramienta integra tres componentes de software: gmond, gmetad, y gweb. En la siguiente figura (Fig-9), se puede observar el esquema del flujo de datos entre ellos, de cómo interactúan, y dan solución de información sobre el estado de recursos computacionales.

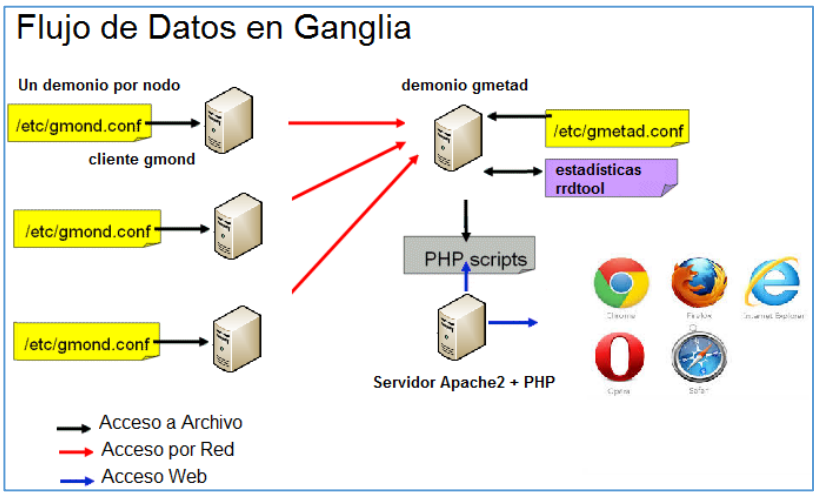


Fig-9

Componente gmond

Es un servicio ligero que se debe instalar en cada nodo de computación del cual se quiere obtener indicadores de funcionamiento. Este demonio lleva a cabo la recolección de métricas reales en cada host mediante un sencillo protocolo de escucha /anuncio para compartir los datos que recolecta con sus nodos pares del clúster. Es posible obtener métricas del sistema, tales como el uso del CPU, memoria, disco, red, y datos sobre los procesos activos.

Internamente, gmond tiene un diseño modular, en forma de pequeños plug-ins específicos del sistema operativo escritos en C para tomar métricas. En Linux, por ejemplo, el plug-in CPU consulta el sistema de ficheros "/proc". Sólo los plugins necesarios están instalados en tiempo de compilación, y gmond tiene, como resultado, un modesto tamaño y sobrecarga insignificante para el Sistema Operativo, en comparación con agentes de supervisión tradicionales. Este componente permite ampliar a plugins escritos en otros idiomas, incluyendo C, C++ y Python para la inclusión de nuevos indicadores.

En una configuración en clúster, y teniendo en cuenta la posibilidad de armar colas de trabajo, definiendo como ‘cola de trabajo’ a la organización de nodos de computación con capacidades similares, en sub-grupos identificables por un nombre, uno de esos nodos concentrará la información recolectada por gmond de los demás miembros. Esta información será a su vez suministrada al demonio gmetad que, generalmente, corre en el head-node del clúster.

Este diseño tiene consecuencias positivas para la escalabilidad global y la capacidad de recuperación del sistema. Sólo un nodo por clúster necesita ser consultado para recopilar todo el estado de ese sub-grupo de clúster.

Componente gmetad

Es el demonio que corre en un host del sistema en clúster, es común ubicarlo en el host head-node. Este componente se ocupa de obtener la información que los nodos gmond recolectan de cada uno de los nodos de cálculo.

Toda la información que obtiene **gmetad**, es almacenada en una base de datos RRDtool, y es por ese motivo que se incluye este punto en el presente trabajo. Se puede decir que la distribución Ganglia ha optado por esta solución de gestión de datos para la registración eficiente de datos de serie de tiempo, y en particular para la generación de gráficos de muy alta calidad visual. [Fig-10]

Las métricas obtenidas por **gmetad,** se almacenan en bases de datos "round robin". Si consultamos nuestros datos cada 10 segundos, por ejemplo, por un valor de un solo día de estas medidas requeriría el almacenamiento de 8.640 puntos de datos. Esto está bien para un par de días de datos, pero no es óptimo para almacenar 8.640 puntos de datos por día durante un año por cada métrica en cada nodo del clúster.

Este componente tiene otras características interesantes, incluyendo la capacidad de sondear datos de otras instancias gmetad, lo que permite la creación de, por ejemplo, una federación de arquitectura jerárquica. Incluye también funcionalidad de consulta interactiva y puede ser consultado por los sistemas de monitoreo externos a través de un protocolo simple de texto en el puerto TCP 8652.

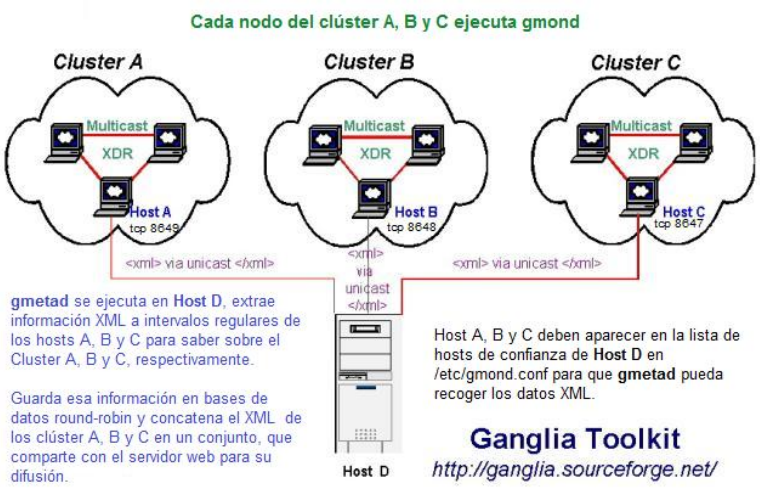


Fig-10

Componente gweb

Como en todo sistema que por objeto tiene la medición de variables, sin duda el componente de visualización de esos datos toma importancia relevante.

Se compone de un programa PHP, que corre bajo cualquier servidor web con soporte PHP o FastCGI. Se instala normalmente en los nodos de cálculo, al igual que gmetad, porque necesita el acceso a las bases de datos RRD generadas por el consultante.

Está organizado en una serie de pestañas de nivel superior: Principal, Búsqueda, Vistas, Agregada gráficos, Comparación de Host, Eventos, rotación automática,

Dashboard Live y Mobile. Estas pestañas permiten saltar fácilmente a la información requerida.

La pestaña principal de gweb, se organiza en torno a conceptos básicos: redes, clústeres, y nodos.

La vista Grid (ver Fig-11) proporciona la vista de nivel más alto disponible, sus gráficos resumen los de datos de todos los hosts conocidos por un único proceso **gmetad**. Es posible, a partir de esta vista, saltar a navegar en detalles de agrupamientos o colas y los hosts que componen esos grupos.

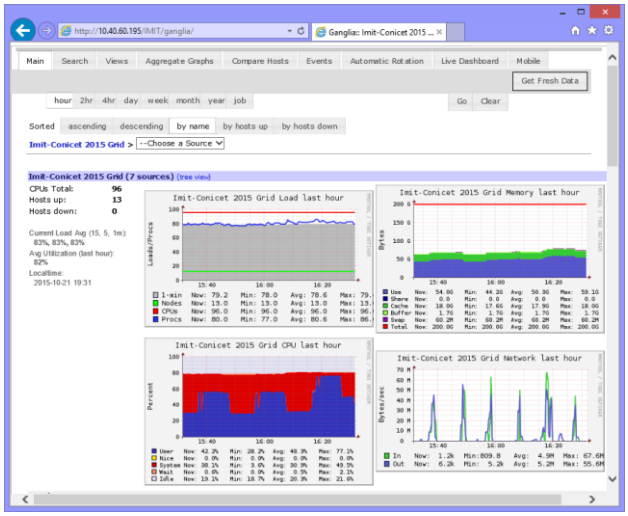


Fig-11

La Vista Clúster.

Para Ganglia, un clúster es una colección nodos de cálculo que ejecutan **gmond** y reportan sus datos a un host determinado, que corre el demonio **gmetad**. Los nodos de un clúster pueden ser agrupados por ubicación física, por carga de trabajo asignada, por la misma capacidad de proceso: en número de cores y de memoria RAM. Al seleccionar uno de los agrupamientos, el sistema mostrará información en resumen de todos los nodos que lo componen, más una vista rápida de cada host individual más abajo en la página.

En la siguiente figura (Fig-12) observamos un ejemplo, con la selección de cola 'Paralela' que integra 4 hosts y 16 cores:

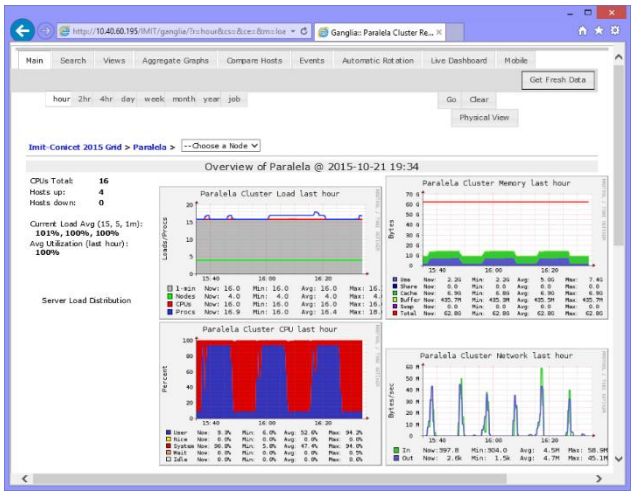


Fig-12

La vista Clúster proporciona una visualización alternativa conocida como vista física, muy útil para grupos muy numerosos de nodos. En la siguiente figura (Fig-13) observamos la vista hosts, del clúster Paralela, y métrica seleccionada: 'mem_cached':

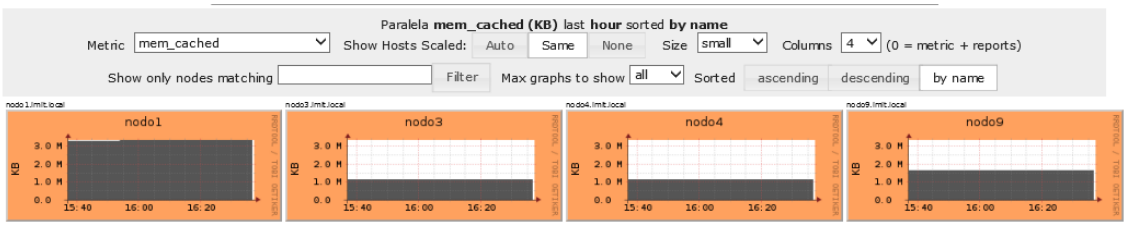


Fig-13

Es posible obtener una visión resumida de la información de un clúster, de sólo texto (Fig-14). Al omitir imágenes, se logra rápidamente la vista principal del clúster.

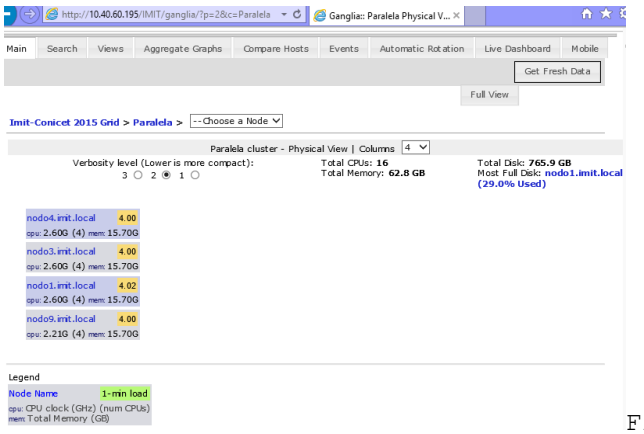


Fig-14

Vista Host.

En esta vista, se muestran y se resumen métricas de un único proceso **gmond,** es decir de un único host. Un resumen de gráficos se muestran en la parte superior, y métricas individuales se agrupan más abajo, ver figura (Fig-15).

Se observa información textual sobre el host, incluida la última vez que realizó el arranque, la versión del sistema operativo y del kernel, el tiempo en funcionamiento (uptime), uso en porcentaje de memoria swap, entre otros.

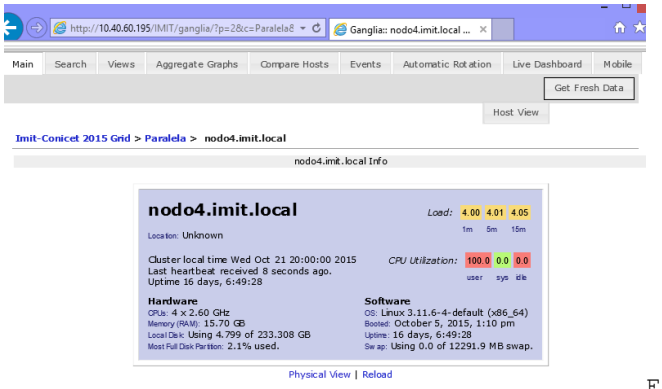


Fig-15

4.8.2. Métricas de base

Desde el lanzamiento de Ganglia, **gmond** fue diseñado para recoger decenas de métricas del sistema que incluía valores tomados de CPU, de memoria, de disco, de la red, y aquellos relacionados con los procesos. Este conjunto de métricas tenía la particularidad de ser 'no modificable', y se las conocía como 'las mediciones default' o 'de base', que la mayoría de los sistemas de seguimiento utilizan para la recolección. Ver Tabla (Tab-2).

Una de las ventajas de apoyar a un conjunto fijo de métricas era que **gmond** podría construirse como un demonio de recopilación de métricas autónomo muy simple. El ejecutable de gmond fue muy pequeño y de poco impacto sobre el sistema al momento de su ejecución.

4.8.3. Instalación de Ganglia

Como se mencionó al principio de este punto, Ganglia integra tres componentes: gmond, gmetad y gweb. El proceso de instalación es muy sencillo, y los requerimientos de bibliotecas, se instalan por defecto en la mayoría de las distribuciones modernas de Linux, y generalmente contienen a libconfuse, pkgconfig, PCRE, y APR.

Los paquetes de Ganglia están disponibles para la mayoría de las distribuciones de Linux, por lo que si opta por usar el gestor de paquetes que vienen con la distribución utilizada, y que es la sugerencia por omisión para su instalación, no deberían aparecer inconvenientes en la resolución de dependencias.

Para gweb en particular, se debe tener instalado de manera previa los siguientes componentes:

- Servidor Web Apache
- PHP 5.2 o posterior
- Extensión PHP JSON instalado y habilitado

La instalación de test en una máquina virtual se realizó con Sistema Operativo OpenSUSE versión 13.1, con 1GB RAM, 1 Núcleo Intel Core 2 T4400. Los siguientes archivos fueron bajados desde Internet e instalados mediante rpm:

```
# rpm –Uhi libganglia-3_1_0-3.1.1.1932-4.1.x86_64.rpm
# rpm –Uhi libpng3-1.2.39-2.4.1.x86_64.rpm
# rpm –Uhi ganglia-gmond-3.1.1-5.3.x86_64.rpm
# rpm –Uhi ganglia-gmetad-3.1.1.1932-4.1.x86_64.rpm
# rpm –Uhi ganglia-web-3.6.2-2.1.noarch.rpm
```

Como requerimiento, es necesario contar con el paquete LAMP (Apache, Mysql, Php); y las librerías libxml2 y libconfuse.

El proceso de instalación genera una carpeta '/etc/ganglia' que básicamente aloja 2 archivos de configuración: 'gmond.conf', que guarda parámetros que tienen que ver la recolección de datos, la configuración IP/UDP necesaria para pasar esos datos; y el 'gmetad.conf', que guarda parámetros para la recepción de datos desde los procesos gmond de los hosts vecinos.

Las opciones de configuración del archivo gmond.conf que se necesitan especificar son:

```
globals {
   daemonize = yes
   setuid = yes
   user = nobody
   debug_level = 0
   max_udp_msg_len = 1472
   mute = no
   deaf = no
   allow_extra_data = yes
   host_dmax = 86400
   cleanup_threshold = 300 /*secs */
   gexec = no
   send_metadata_interval = 30 /*secs */
 }
 cluster {
   name = " RRDtool-Test-2015"
   owner = "UNNE"
   latlong = "unspecified"
   url = "unspecified"
 }
 host {
   location = "unspecified"
 }
 udp_send_channel {
   host = 192.168.194.88
   port = 8649
   ttl = 1
 }
 udp_recv_channel {
   port = 8649
 }
```

```
tcp_accept_channel {
  port = 8649
}
```

Estos parámetros básicos, corresponden a una infraestructura basada en red TCP-IP Unicast, esta configuración se debe copiar a todos los nodos de cálculo para los que se desea recopilar información.

Las opciones de configuración del archivo gmetad.conf que se necesitan modificar son:

```
# nombre de la BD, IP y puerto TCP del origen de los datos
data_source "RRDtool-Test-2015" 127.0.0.1:8649
# Lugar donde gmetad guarda sus archivos round-robin
default: "/var/lib/ganglia/rrds"
```

La primer línea declara el nombre con el cual serán generadas las bases de datos RRD's, la dirección IP desde el cual llegarán los sondeos de datos y el número de puerto TCP asociado. Para configuraciones de múltiples nodos a sondear, serán necesarios números de puerto TCP distintos.

La segunda línea, especifica el lugar donde los archivos de RRD's serán almacenados.

La siguiente figura (Fig-16) resume la configuración inicial del servicio web y de RRDtool.

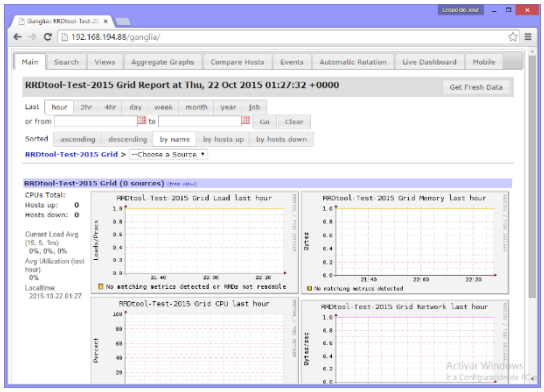


Fig-16

Es posible apreciar en la figura (Fig-16) que faltan configurar los accesos a los nodos, y el reinicio del servicio para obtener visualización de sus datos.

5. Conclusiones

RRDtool es una herramienta construida sobre el concepto de Round-Robin Database. Trata de un tipo muy específico de base de datos, orientadas al almacenamiento de datos basados en series temporales, y que garantizan el espacio de almacenamiento final ocupado por sus elementos.

Fue posible demostrar propiedades y características que se consideran de provecho y que lo califican positivamente como manejador, gestor y graficador de datos de series de tiempo, recogidos en un ambiente HPC.

Para quienes hacen desarrollo, testing de software; como también administración de los sistemas de computación, resulta importante conocer las lecturas de variables del sistema que han sido registradas con las técnicas descritas, durante la ejecución del código. Esto permite, con el análisis de los datos obtenidos, mejorar el conocimiento sobre el uso de los recursos (memorias caché, memoria RAM), y determinar en consecuencia, las partes del código que podrían ser mejorados para su optimización.

Al momento de cerrar este informe, la herramienta RRDtool permanece en el primer puesto de un listado alojado en el sitio en Internet “db-engines.com” [1], que como aspecto fundamental, mide su grado de popularidad. Esta clasificación no mide el número de instalaciones de los sistemas, o de su uso dentro de los sistemas de TI. Optan por obtener datos aportados por un aumento de la popularidad de un sistema, por ejemplo, en las discusiones técnicas o en las ofertas de empleo.

5.1. Trabajo a futuro

A partir de esta aproximación a la herramienta RRDtool, es posible abordar el desarrollo de un proyecto de software que ayude, por ejemplo, a los usuarios que desarrollan, hacen testing y administración de aplicaciones de cálculo de alto rendimiento, a formular un modelo de datos en la modalidad vista.

El trabajo podría integrar una manera de seleccionar variables del sistema de cómputo que se desea monitorear en profundidad, en función al tipo de ejecución (memoria compartida, memoria distribuida). Al final de la selección, se tendrá definido una especie de template que ayudará a determinar la configuración de los archivos RRD necesarios, más un script que defina la recolección de datos, y un sitio web para su posterior visualización.

Estas primeras experiencias en la generación y monitoreo de bases de datos de serie de tiempo, como también del conocimiento adquirido sobre el escenario donde es posible montar estas tecnologías como ser clúster de servidores de cálculo; podrán constituirse en un aporte a la comunidad científica para el monitoreo y seguimiento de trabajos a distancia.

6. Referencias y Bibliografía

[1] http://db-engines.com/en/ranking/time+series+dbms

[2] Monitoring with Ganglia - ISBN: 978-1-449-32970-9

[3] https://tobi.oetiker.ch/hp/about/

[4] http://www.gnu.org/licenses/licenses.es.html

[5] http://oss.oetiker.ch/mrtg/doc/mrtg-rrd.en.html

[6] https://www.nagios.org/

[7] https://collectd.org/

[8] http://nmon.sourceforge.net/pmwiki.php

[9] http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html

[10] http://www.supercalculo.mincyt.gob.ar/

[11] https://calomel.org/rrdtool.html

[12] http://oss.oetiker.ch/rrdtool/doc/rrdtool.en.html

[13] http://blog.desdelinux.net/cron-crontab-explicados

[14] https://tools.ietf.org/html/rfc1157

[15] http://rrdtool.vandenbogaerdt.nl/min-avg-max.php

[16] http://oss.oetiker.ch/rrdtool/tut/cdeftutorial.en.html

[17] http://oss.oetiker.ch/rrdtool/tut/rrd-beginners.en.html

[18] http://www.fromdual.com/sites/default/files/rrd.pdf

[19] http://oss.oetiker.ch/rrdtool/download.en.html

[20] https://www.otexts.org/fpp/7/5

[21] http://sourceforge.net/p/ganglia/wiki/Home/

ANEXO TABLAS

TAB-1. Lista de los archivos de base de datos (RRD's) activas en un nodo de cálculo

```
newmanager:/var/lib/ganglia/rrds/Nodo12/nodo12 # ls
boottime.rrd
cpu_idle.rrd
cpu_system.rrd
disk_total.rrd
mem_buffers.rrd
mem_total.rrd
proc_run.rrd
bytes_in.rrd
cpu_nice.rrd
cpu_user.rrd
load_fifteen.rrd
mem_cached.rrd
part_max_used.rrd
proc_total.rrd
bytes_out.rrd
cpu_num.rrd
cpu_wio.rrd
load_five.rrd
mem_free.rrd
pkts_in.rrd
swap_free.rrd
cpu_aidle.rrd
cpu_speed.rrd
disk_free.rrd
load_one.rrd
mem_shared.rrd
pkts_out.rrd
swap_total.rrd
```

TAB-2. Lista de métricas obtenidas en un host por el proceso gmond

```
# gmond -m
mem_shared      Amount of shared memory (module mem_module)
proc_total      Total number of processes (module proc_module)
pkts_in         Packets in per second (module net_module)
multicpu_nice0   Percentage of CPU utilization that occurred while executing at
the nice level (module multicpu_module)
multicpu_user0   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
os_release      Operating system release date (module sys_module)
multicpu_nice1   Percentage of CPU utilization that occurred while executing at
the nice level (module multicpu_module)
multicpu_user1   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
multicpu_nice2   Percentage of CPU utilization that occurred while executing at
the nice level (module multicpu_module)
multicpu_user2   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
multicpu_nice3   Percentage of CPU utilization that occurred while executing at
the nice level (module multicpu_module)
multicpu_user3   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
cpu_intr         (module cpu_module)
multicpu_nice4   Percentage of CPU utilization that occurred while executing at
the nice level (module multicpu_module)
multicpu_user4   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
multicpu_nice5   Percentage of CPU utilization that occurred while executing at
```

```
the nice level (module multicpu_module)
multicpu_user5   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
cpu_aidle       Percent of time since boot idle CPU (module cpu_module)
multicpu_nice6   Percentage of CPU utilization that occurred while executing at
the nice level (module multicpu_module)
multicpu_user6   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
proc_run        Total number of running processes (module proc_module)
multicpu_nice7   Percentage of CPU utilization that occurred while executing at
the nice level (module multicpu_module)
multicpu_user7   Percentage of CPU utilization that occurred while executing at
the user level (module multicpu_module)
mem_buffers     Amount of buffered memory (module mem_module)
part_max_used   Maximum percent used for all partitions (module disk_module)
heartbeat       Last heartbeat (module core_metrics)
multicpu_wio0    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
multicpu_wio1    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
multicpu_wio2    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
multicpu_wio3    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
multicpu_wio4    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
multicpu_wio5    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
mem_cached      Amount of cached memory (module mem_module)
multicpu_wio6    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
cpu_num         Total number of CPUs (module cpu_module)
multicpu_wio7    Percentage of CPU utilization that occurred while executing at
the wio level (module multicpu_module)
cpu_speed       CPU Speed in terms of MHz (module cpu_module)
location        Location of the machine (module core_metrics)
boottime        The last time that the system was started (module sys_module)
load_five       Five minute load average (module load_module)
cpu_system       Percentage of CPU utilization that occurred while executing at
the system level (module cpu_module)
mtu             Network maximum transmission unit (module sys_module)
machine_type    System architecture (module sys_module)
cpu_sintr       cpu_sintr (module cpu_module)
pkts_out        Packets out per second (module net_module)
bytes_out       Number of bytes out per second (module net_module)
load_one        One minute load average (module load_module)
os_name         Operating system name (module sys_module)
multicpu_system0             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
multicpu_system1             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
multicpu_system2             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
sys_clock       Time as reported by the system clock (module sys_module)
multicpu_system3             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
multicpu_system4             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
multicpu_system5             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
multicpu_system6             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
multicpu_system7             Percentage of CPU utilization that occurred while
executing at the system level (module multicpu_module)
multicpu_sintr0  Percentage of CPU utilization that occurred while executing at
the sintr level (module multicpu_module)
multicpu_sintr1  Percentage of CPU utilization that occurred while executing at
the sintr level (module multicpu_module)
multicpu_sintr2  Percentage of CPU utilization that occurred while executing at
```

```
the sintr level (module multicpu_module)
multicpu_sintr3  Percentage of CPU utilization that occurred while executing at
the sintr level (module multicpu_module)
multicpu_sintr4  Percentage of CPU utilization that occurred while executing at
the sintr level (module multicpu_module)
cpu_wio           Percentage of time that the CPU or CPUs were idle during which
the system had an outstanding disk I/O request (module cpu_module)
multicpu_sintr5  Percentage of CPU utilization that occurred while executing at
the sintr level (module multicpu_module)
multicpu_sintr6  Percentage of CPU utilization that occurred while executing at
the sintr level (module multicpu_module)
multicpu_sintr7  Percentage of CPU utilization that occurred while executing at
the sintr level (module multicpu_module)
swap_free        Amount of available swap memory (module mem_module)
bytes_in         Number of bytes in per second (module net_module)
swap_total       Total amount of swap space displayed in KBs (module mem_module)
multicpu_intr0   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_intr1   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_idle0   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
mem_free         Amount of available memory (module mem_module)
load_fifteen     Fifteen minute load average (module load_module)
multicpu_intr2   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_idle1   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
multicpu_intr3   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_idle2   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
mem_total        Total amount of memory displayed in KBs (module mem_module)
multicpu_intr4   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_idle3   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
multicpu_intr5   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_idle4   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
multicpu_intr6   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_idle5   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
cpu_idle          Percentage of time that the CPU or CPUs were idle and the system
did not have an outstanding disk I/O request (module cpu_module)
multicpu_intr7   Percentage of CPU utilization that occurred while executing at
the intr level (module multicpu_module)
multicpu_idle6   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
cpu_user           Percentage of CPU utilization that occurred while executing at
the user level (module cpu_module)
cpu_nice           Percentage of CPU utilization that occurred while executing at
the user level with nice priority (module cpu_module)
multicpu_idle7   Percentage of CPU utilization that occurred while executing at
the idle level (module multicpu_module)
disk_free        Total free disk space (module disk_module)
gexec            gexec available (module core_metrics)
disk_total       Total available disk space (module disk_module)
```

TAB-3 – Ejercicio práctico, comandos RRDtool.

El siguiente ejemplo fue tomado del sitio en línea de referencia [11], y tiene por objeto mostrar el funcionamiento de la herramienta RRDtool, a través de sus comandos y funcionalidades incorporadas.

Una base de datos es generada con la siguiente estructura:

```
rrdtool create latency_db.rrd \
--step 60 \
DS:pl:GAUGE:120:0:100 \
DS:rtt:GAUGE:120:0:10000000 \
RRA:MAX:0.5:1:1500 \
RRA:MIN:0.5:1:1500
```

Se registrarán la pérdida de paquetes (variable 'pl') y tiempo de ida y vuelta (variable 'rtt') de paquetes de datos enviados hacia un host determinado en la Internet.
El parámetro '--step 60' nos dice el tiempo en segundos en que esperamos que los datos se actualizan en la base de datos.

De la línea '*DS:pl:GAUGE:120:0:100* ', tenemos:

- DS dice que se trata de un conjunto de datos.
- pl es el nombre de la variable que hemos elegido para representar "pérdida de paquetes".
- GAUGE indica que los datos introducidos son absolutos y deben introducirse sin ningún tipo de manipulación o cálculo hechos sobre él.
- 120 es el tiempo de espera en segundos. Si no se introducen datos en al menos 120 segundos después se insertarán ceros como valores. Puesto que hemos creado una base de datos RRD con un paso de 60 segundos (--step 60) tendríamos que perder 2 "pasos" completos antes de RRDtool complete con ceros. Este tiempo de espera es importante para significar si el sistema no ha podido recopilar datos debido a un reinicio o tiempo de inactividad del sistema. Las pérdidas de datos se mostrarán luego en el gráfico como un área en blanco sin datos graficados.
- 0 es el valor mínimo que será aceptado en la base de datos. Puesto que la variable es para la pérdida de paquetes se espera que el valor sea entre el 0% y el 100%.
- 100 es el valor máximo que se acepta en este campo. Esta variable es la pérdida de paquetes (pl) y esperamos ver un valor entre 0% y 100%.

De la línea '*DS:rtt:GAUGE:120:0:10000000*', tenemos:

- DS dice que se trata de un conjunto de datos.
- rtt es el nombre de la variable que hemos elegido para representar el "tiempo total del viaje".
- GAUGE indica que los datos introducidos son absolutos y deben introducirse sin ningún tipo de manipulación o cálculo hechos sobre él.
- 120 es el tiempo de espera en segundos. Si no se introducen datos en al menos 120 segundos después se insertarán ceros como valores. Puesto que hemos creado una base de datos RRD con un paso de 60 segundos (--step 60) tendríamos que perder 2 "pasos" completos antes de RRDtool complete con ceros. Este tiempo de espera es importante para significar si el sistema no ha podido recopilar datos

debido a un reinicio o tiempo de inactividad del sistema. Las pérdidas de datos se mostrarán luego en el gráfico como un área en blanco sin datos graficados.

- 0 es el valor mínimo que será aceptado en la base de datos. Puesto que la variable es para la pérdida de paquetes se espera que el valor sea entre el 0% y el 100%.
- 10000000 es el valor máximo que se acepta en este campo. Es sólo un valor suficiente como entender que se aceptará cualquier valor RTT de gran tamaño.

De la línea '*RRA:MAX:0.5:1:1500*', tenemos:

- La entrada RRA define cuántos valores de la base de datos de la RRD archivará y por cuánto tiempo.
- MAX significa que aceptará sólo el valor máximo si varios valores están disponibles a través de múltiples "pasos". Estamos utilizando MAX simplemente para decir que tenemos una variable que contendrá un número o un promedio y no debe ser modificado de ninguna manera.
- 0,5 es un valor interno de resolución, conocido como factor 'xff'.
- 1 especifica cuántos pasos deben ser evaluados antes de almacenar el valor final. Se especifica "1" porque queremos que el valor actualizado en la base de datos para almacenar tal cual; un paso es igual a un valor de base de datos.
- 1500 es el número de "registros" a almacenar en la base de datos. Puesto que especificamos un paso de 60 segundos (--step 60) almacenaremos 1500 muestras por 60 segundos lo que equivale a 90.000 segundos. Esto también es igual a 25 horas. Por lo tanto, vamos a tener 25 horas de datos de una resolución de 1 minuto que podemos representar gráficamente. Esta es una buena granularidad y nos permitirá hacer gráfico preciso y de muy buen aspecto.

De la línea '*RRA:MIN:0.5:1:1500*', tenemos:

- Ídem al anterior, con la diferencia en que MIN guardará un valor mínimo si varios valores están disponibles a través de múltiples "pasos".

Ahora, el siguiente script es utilizado para introducir datos en la base de datos. Básicamente se generan procesos 'ping' hacia un destino en Internet, registrando en variables los valores 'pl' y 'rtt' obtenidos:

```
#!/usr/local/bin/bash
#
### set the paths
command="/sbin/ping -q -n -c 3"
gawk="/usr/local/bin/gawk"
rrdtool="/usr/local/bin/rrdtool"
### Direccion IP del DNS de Google publico
hosttoping="8.8.8.8"

### data collection routine
get_data() {
    local output=$($command $1 2>&1)
    local method=$(echo "$output" | $gawk '
        BEGIN {pl=100; rtt=0.1}
        /packets transmitted/ {
            match($0, /([0-9]+)% packet loss/, datapl)
            pl=datapl[1]
        }
        /min\/avg\/max/ {
            match($4, /(.*)\/(.*)\/(.*)\/(.*)/, datartt)
            rtt=datartt[2]
```

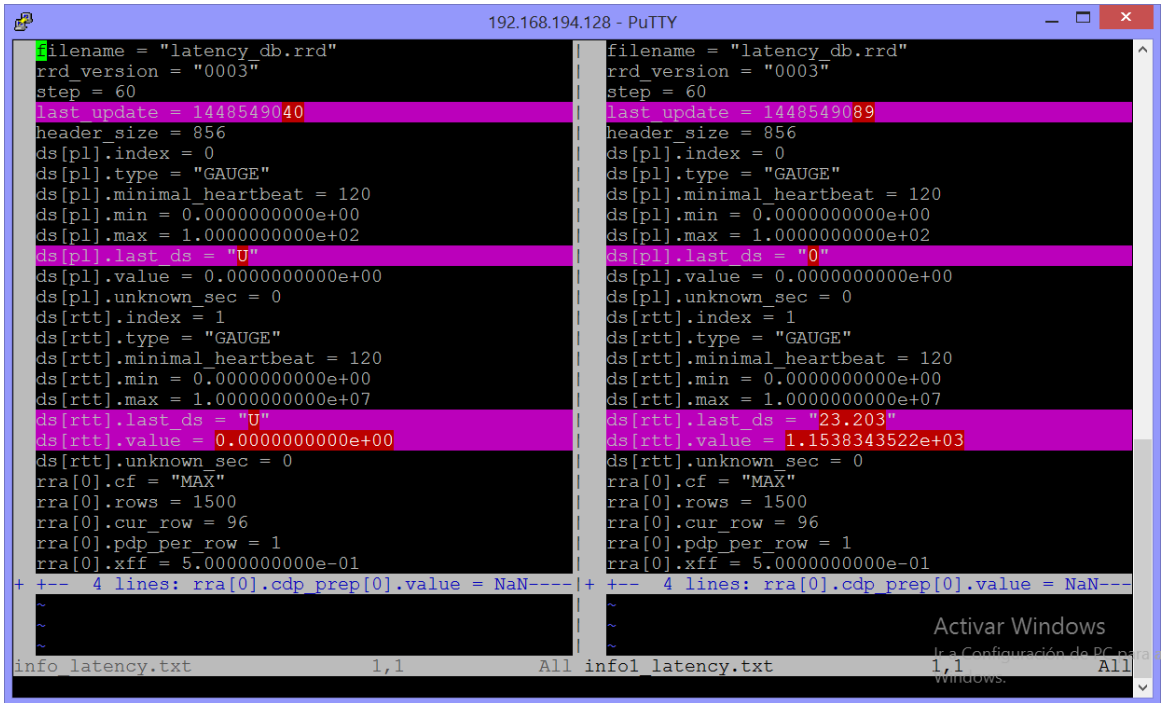
```
        }
        END {print pl ":" rtt}
        ')
    RETURN_DATA=$method
}
### recolecta datos
get_data $hosttoping

### actualiza la base de datos
$rrdtool update latency_db.rrd --template pl:rtt N:$RETURN_DATA
```

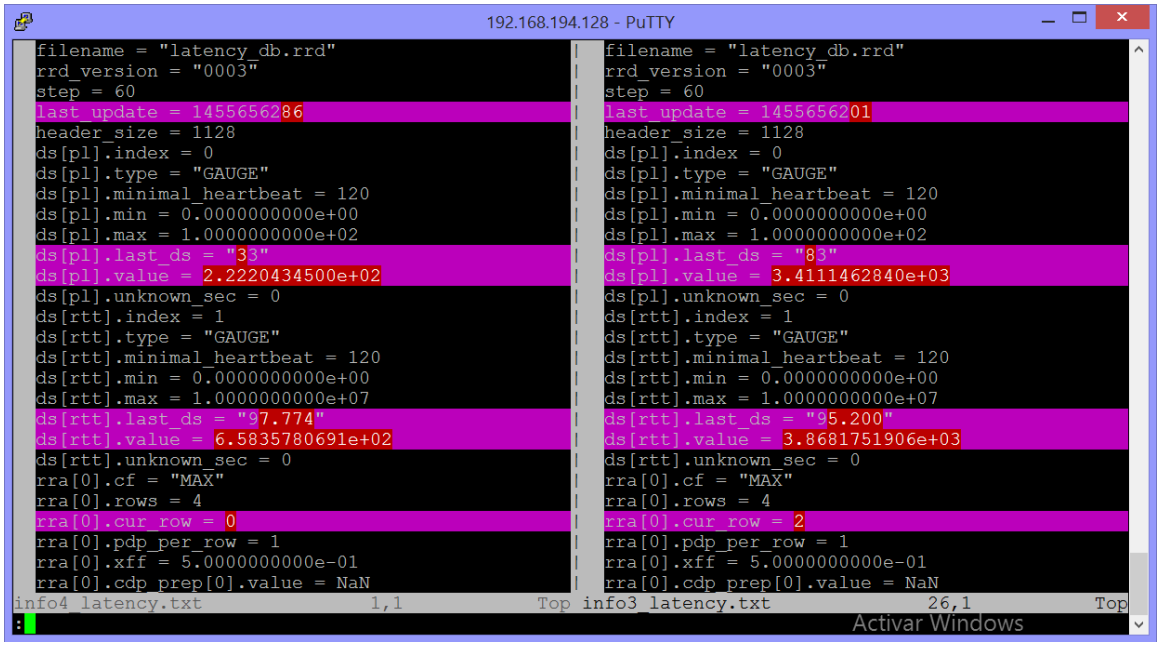
La automatización en la recolección de datos, puede ser realizada mediante 'cron' del sistema, introduciendo una entrada que ejecute el script anterior.

Antes de cualquier actualización, se genera una salida en formato de texto con el comando '$ rrdtool info latency_db.rrd > info_latency.txt', a los efectos de observar el estado de variables de la base de datos. Luego de una actualización se repite el comando pasando a 'info1_latency.txt', a 'info2_latency.txt', 'info3_latency.txt' en actualizaciones subsiguientes.

A partir de los archivos de texto generados, deseamos mostrar a través de una simple comparativa, los valores que aloja la base datos a medida que está siendo actualizada.
Utilizamos para ello el programa 'vimdiff' de Linux, que muestra en color rojo resaltado, las diferencias que encuentra entre los archivos:



En la imagen de arriba, se observa las diferencias entre los dos archivos iniciales. Sobre la izquierda, el contenido de 'info_latency.txt', y a la derecha, 'info1_latency.txt' luego de la primer actualizacion.
Para el primer caso y como ejemplo, el parametro 'ds[rtt].last_ds' es igual a 'U', o 'unknow' desconocido, dado que no tiene aun valores guardados. Para el caso del segundo archivo, se observa el valor '23.203' que se corresponde al tiempo total del viaje del proceso ping que ha sido registrado.



En esta imagen, y para el lado izquierdo, archivo 'info4_latency.txt', un valor de '33' para el 'ds[pl].last_ds' 'pl'; y un valor de '83' para el lado derecho. Representan pérdida de paquetes en porcentajes. Asimismo, para el valor 'ds[rtt].last_ds', en la izquierda, registra un tiempo de 97.774 ms contra '95.200' de la derecha.

Es posible consultar mediante la herramienta rrdtool obtener información acerca del estado de la base datos:

- Datos de la primer actualización:

```
leopoldo@miLinux:~/especializa/ping2> rrdtool first latency_db.rrd

1455656280
```

Y nos devuelve vacío, por no contener datos.

- Datos de la última actualización:

```
leopoldo@miLinux:~/especializa/ping2> rrdtool lastupdate latency_db.rrd
             pl    rtt

1455656466: 0     103.146
```

Comentario: no hubo pérdida de paquetes en la última actualización.

- Datos de una actualización mas reciente:

```
leopoldo@miLinux:~/especializa/ping2> rrdtool lastupdate latency_db.rrd
             pl    rtt

1455673147: 16    30.874
```

Comentario: para este caso, se observa que en la última recolección de datos, el proceso ping obtuvo 16% de pérdida de paquetes.

LISTA DE FIGURAS.