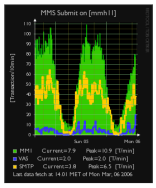
rrdtool: scripting & graphs

Controllare il proprio sistema con grafici automatizzati



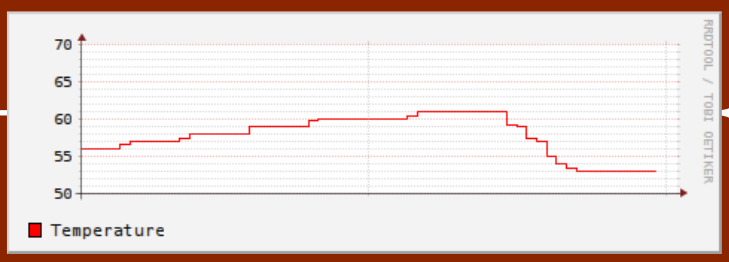
Capita a tutti gli smanettoni di passare nottate e nottate intere a consultare log, a spulciare lunghi file di testo in cerca di qualche informazione interessante. Sebbene guardare una sfilza di caratteri bianchi su sfondo nero possa esser piacevole, a volte un'immagine "vale piu' di cento parole" (slogan preso in prestito da qualcun'altro). Ecco in cosa son utili gli **rrdtool** [0]. Con gli rrdtool e' possibile scrivere dei semplici script (bash, python, ruby, lua ecc...) per monitorare con dei grafici quello che vi pare. Il classico esempio che si trova girando un po' per la rete e' uno script per monitorare l'uso della cpu e della memoria. Semplice, ma non per questo inutile. Un altro semplice esempio (nonche' lo script che ho realizzato io per prendere confidenza con lo strumento) e' dato dal controllo della temperatura del laptop. Con rrdtool ho realizzato due piccoli script per tracciare il grafico dell'evoluzione temporale della temperatura in condizioni di stress della scheda video... Davvero nulla di complicato con questo comodo tool...

rrdtool

Cominciamo a vedere in cosa consistono gli rrdtool, e prima ancora cos'e' un **rrd**. La sigla sta per *Round Robin Database*, come ben potrete capire, un database! Per la precisione un database a dimensione fissa. A che serve utilizzare un database in uno script che traccia l'andamento della temperatura? Lo script va a memorizzare nel database ogni misura effettuata, sovrascrivendo, una volta terminati i posti liberi, le prime misure inserite (il database quindi non crescera' a dismisura). rrdtool andra' a *pescare* dal database i record per poi tracciare il grafico secondo le direttive da noi inserite.
Mettiamo in pratica quanto detto fin'ora con il semplicissimo esempio citato. Realizziamo dunque due script che chiameremo *riempi-database.sh* e *traccia-grafico.sh*.
Vediamo e commentiamo il primo (listato 0). Ok, cerchiamo di capire cosa abbiamo scritto:
La prima riga e' il classico *sha*-bang, uno speciale commento che istruisce il nostro sistema operativo circa l'interprete da usare per interpretarlo. Passiamo a qualcosa di piu' misterioso... La seconda riga invoca rrdtool per creare il database thermal.rrd. In questo database vogliamo inserire dati ogni 60 secondi (*--step 60*) e vogliamo un solo campo (*temperature*). Questa variabile da memorizzare nell'archivio e' di tipo *GAUGE*, cioe' si tratta di un valore assoluto e non relativo a precedenti misurazioni. Ogni volta che si acquisisce una lettura siamo disposti a tollerare 15 secondi di ritardo, e siamo disposti ad accettare un valore che sia compreso fra 0 e 100. Analizziamo ora l'ultima parte della seconda riga. In questa andremo a descrivere l'archivio (RRA sta infatti per *Round Robin Archive*) anziche' i suoi campi.
Il campo impostato a 1 ci permette di scegliere ogni quanti valori acquisiti memorizzare informazioni nel database, ad esempio se impostassimo 5 per ogni 5 letture avremmo un'inserimento nel database. Ok, ma cosa viene inserito? Il primo valore letto fra quei cinque? O l'ultimo? Nel nostro caso avremmo un valor medio data la parola chiave AVERAGE. 60 a fine riga indica i posti disponibili nel database (quindi per un rapido calcolo se facciamo una misura al minuto e di misure ne facciamo 60 avremo l'andamento della temperatura nell'ultima ora...). Ok, ci rimane da descrivere un campo, quello contenente il valore 0.5... Cosa vorra' dire? Indica la percentuale di dati ignoti accettabili nel database.
Questa era la parte piu' complicata dello script. Segue un semplicissimo ciclo while (infinito) che memorizza nella variabile $temp la temperatura rilevata e la inserisce nel database tramite rrdtool. I parametri passati a rrdtool sono *update* (per aggiornare l'archivio), *thermal.rrd* (che indica il file da aggiornare), *N:${temp}* che informa rrdtool che la misura e' stata effettuata nell'orario corrente *(N)* e che il valore da inserire e'

listato 0

```
#!/bin/bash
rrdtool create thermal.rrd --step 60 DS:temperature:GAUGE:15:0:100 RRA:AVERAGE:0.5:1:60
while [ 1 ]
do
        temp=$(cat /proc/acpi/thermal_zone/THM0/temperature | grep '[0-9][0-9]' --only-matching)
        rrdtool update thermal.rrd N:${temp}
        sleep 60
done
```

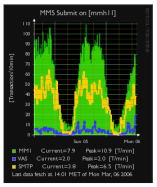


listato 1

```
#!/bin/bash
rrdtool graph graph.png DEF:tmp=thermal.rrd:temperature:AVERAGE\  LINE1:tmp#ff0000:Temperatura –start
-n1 --end N
display graph.png
```

rrdtool: scripting & graphs

Controllare il proprio sistema con grafici automatizzati



quello della variabile $temp.

E il primo script e' fatto... Serve soltanto a misurare la temperatura ogni 60 secondi. Vediamo ora come si traccia il grafico. Basta una *semplice* invocazione di rrdtool che per comodita' abbiamo inserito in uno script (piu' che altro in previsione di future evoluzioni). Leggiamo il listato 1.
Analizziamo e commentiamo anche quest'ultimo script. Tralasciamo l'ormai noto sha-bang e concetriamoci su rrdtool. Vogliamo che il nostro strumento tracci il grafico, quindi il comando che utilizzeremo con rrdtool e' *graph*. *graph.png* e' ovviamente il nome del file immagine contenente il grafico. Con *DEF:tmp=thermal.rrd:temperature:AVERAGE* abbiamo definito una variabile *tmp* che rrdtool utilizzera' per tracciare il grafico. A questa abbiamo assegnato il valore del campo *temperature* dell'archivio *thermal.rrd*. La parola chiave *AVERAGE* indica la funzione *di consolidamento* usata per archiviare i dati. Con *DEF:tmp=thermal.rrd:temperature:AVERAGE* indichiamo a rrdtool di tracciare la linea relativa alla variabile *tmp* con il colore *#ff0000* e di assegnarle come descrizione *Temperatura*. Terminiamo con gli ultimi due parametri: *--start -1h* indica che il grafico mostra l'andamento a partire da un'ora prima dell'ora corrente, mentre *–end N* indica che la fine del grafico coincide con l'istante attuale (*N* sta per *now*). L'ultima riga ci mostra il grafico :-)

Iniziamo a fare sul serio...
rrdtool e' molto piu' potente di quanto abbiamo potuto apprezzare con questo fin troppo semplice esempio. Passiamo a qualcosa di piu' elaborato. Ho voluto fare uno script che tramite un grafico mi mostrasse l'evoluzione del numero di processi al variare del numero di utenti connessi alla mia GNU/Linux box e al variare del numero di sessioni aperte.
Come al solito l'ho fatto tramite due script; il primo e' nel listato 2, il secondo nel 3.
Dal listato 3 potete vedere perche' preferisco usare uno script anche per il *tracer*. In questo modo posso avere una sintassi piu' chiara anziche' avere un'unica linea enorme...
Adesso tocca a voi e alla vostra fantasia scoprire la potenza di rrdtool!!!

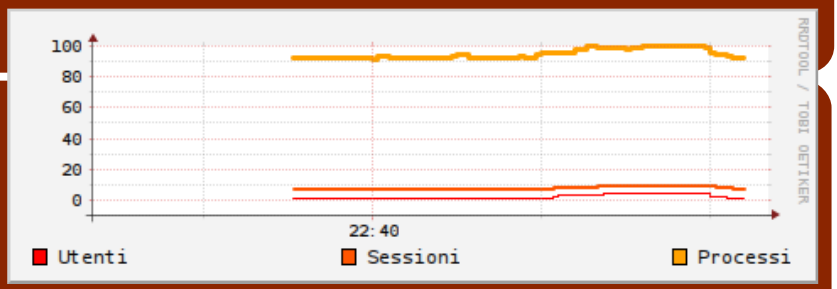
Link e riferimenti:
[0] http://oss.oetiker.ch/rrdtool

listato 2

```
#!/bin/bash

rrdtool create users.rrd --step 1 DS:diffusers:GAUGE:30:0:100 DS:sessions:GAUGE:30:0:100
DS:processes:GAUGE:30:0:100 RRA:AVERAGE:0.5:1:120

while [ 1 ]
do
sessions=$(who | wc -l)
diffusers=$(who | cut -f 1 -d ' ' | uniq | wc -l)
processes=$(ps auxh | wc -l)
rrdtool update users.rrd N:${diffusers}:${sessions}:${processes}
sleep 1
done
```



listato 3

```
#!/bin/bash

DEF1="DEF:u=users.rrd:diffusers:AVERAGE"
DEF2="DEF:s=users.rrd:sessions:AVERAGE"
DEF3="DEF:p=users.rrd:processes:AVERAGE"
LINE1="LINE1:u#ff0000:Utenti"
LINE2="LINE2:s#ff5500:Sessioni"
LINE3="LINE3:p#ffa100:Processi"
AREA1="AREA:u#aecbff"

rrdtool graph graph.png $DEF1 $DEF2 $DEF3 $LINE1 $LINE2 $LINE3 $AREA1 -l -10 --start -0h6m\ --end N -
u 60
display graph.png
```