# Open-Falcon

http://open-falcon.org/

Open-Falcon 是小米运维部开源的一款互联网企业级监控系统解决方案.

社区 | Open-Falcon

<http://book.open-falcon.org/zh/index.html>

测试环境目录

/root/work/open-falcon/agent

开发环境目录

/data

export HOME=/home/work

export WORKSPACE=$HOME/open-falcon

sudo mkdir -p $WORKSPACE

cd $WORKSPACE

mysql -h localhost -u root --password="123456" < db\_schema/graph-db-schema.sql

mysql -h localhost -u root --password="123456" < db\_schema/dashboard-db-schema.sql

mysql -h localhost -u root --password="123456" < db\_schema/portal-db-schema.sql

mysql -h localhost -u root --password="123456" < db\_schema/links-db-schema.sql

mysql -h localhost -u root --password="123456" < db\_schema/uic-db-schema.sql

sudo tar -zxf open-falcon-latest.tar.gz -C ./tmp/

for x in `find ./tmp/ -name "\*.tar.gz"`;do \

app=`echo $x|cut -d '-' -f2`; \

sudo mkdir -p $app; \

sudo tar -zxf $x -C $app; \

done

curl -s "http://127.0.0.1:6060/health"

## Dashboard索引缺失，查询不到endpoint或counter

绘图相关 | Open-Falcon http://book.open-falcon.org/zh/faq/graph.html



## gcc - fatal error: Python.h: No such file or directory - Stack Overflow

http://stackoverflow.com/questions/21530577/fatal-error-python-h-no-such-file-or-directory

Looks like you haven't properly installed the header files and static libraries for python dev. If your OS is Ubuntu/Debian:

sudo apt-get install python-dev

## linux - mysql\_config not found when installing mysqldb python interface - Stack Overflow

http://stackoverflow.com/questions/7475223/mysql-config-not-found-when-installing-mysqldb-python-interface

mySQLdb is a python interface for mysql, but it is not mysql itself. And apparently mySQLdb needs the command 'mysql\_config', so you need to install that first.

Can you confirm that you did or did not install mysql itself, by running "mysql" from the shell? That should give you a response other than "mysql: command not found".

Which linux distribution are you using? Mysql is pre-packaged for most linux distributions. For example, for debian / ubuntu, installing mysql is as easy as

sudo apt-get install mysql

mysql-config is in a different package, which can be installed from (again, assuming debian / ubuntu):

sudo apt-get install libmysqlclient-dev

if you are using mariadb, the drop in replacement for mysql, then run

sudo apt-get install libmariadbclient-dev

Run the following afterwards if sudo apt-get install libmysqlclient-dev still doesn't work

sudo apt-get install python-dev

sudo apt-get install python-mysqldb

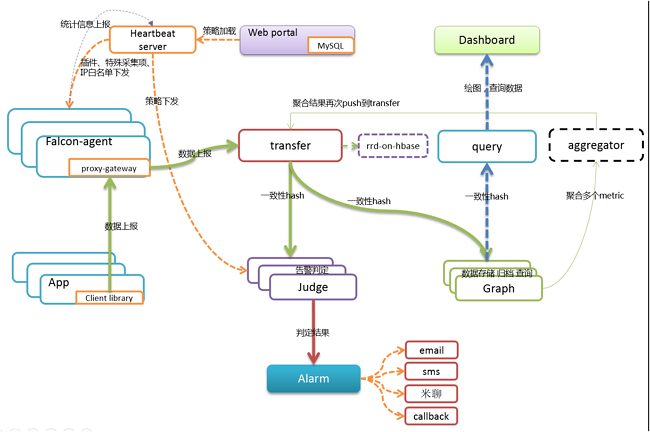
Finally, check if mysql\_config is in your PATH; add if necessary

export PATH=$PATH:/path/to/your/mysql\_config

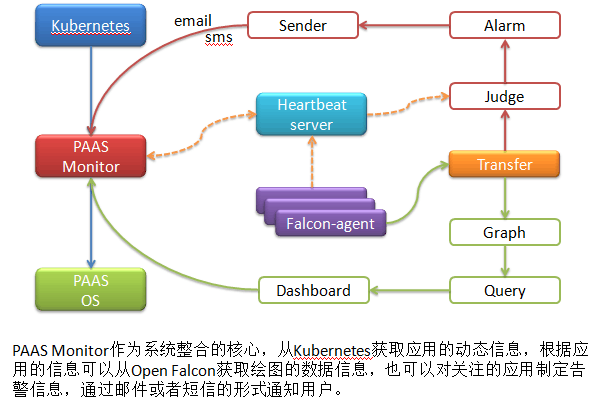
To possibly solve

EnvironmentError: mysql\_config not found

## 架构



## PAAS Monitor架构



## 端口

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **No** | **Module** | **Http** | **Rpc** | **Redis** | **Configure File** | **Language** | **Database** | **Remark** |
|  | fe | 1234 |  | 6379 | cfg.json | go | uic | 用户信息中心User Information Center |
|  | agent | 1988 |  |  | cfg.json | go |  | agent自动采集预先定义的各种采集项，定时(60s)push到transfer |
|  | portal | 5050 |  |  | frame/config.py | python | falcon\_portal | 配置报警策略 |
|  | links | 5090 |  |  | frame/config.py | python | falcon\_links | 当多个告警被合并为一条告警信息时，短信附带告警的http地址 |
|  | hbs | 6031 | 6030 |  | cfg.json | go | falcon\_portal | 每分钟从DB读取一次并缓存到内存，静待agent、judge的请求 |
|  | transfer | 6060 | 8433 |  | cfg.json | go |  | 监听在:8433端口上，agent会通过jsonrpc的方式来push数据上来 |
|  | sender | 6066 |  | 6379 | cfg.json | go |  | 发送由写入redis的报警短信和报警邮件 |
|  | graph | 6071 | 6070 |  | cfg.json | go | graph | 存储绘图数据、历史数据的组件。Transfer将数据转发graph |
|  | judge | 6081 | 6080 | 6379 | cfg.json | go |  | 报警判断模块，judge依赖于HBS |
|  | dashboard | 8081 |  |  | rrd/config.py | python | dashboard, graph | 面向用户的查询界面，用户可以看到graph中的所有数据和趋势图 |
|  | alarm | 9912 |  | 6379 | cfg.json | go |  | judge产生的报警event写入redis，alarm从redis读取 |
|  | query | 9966 |  |  | cfg.json | go |  | 收到用户的查询请求后，查询相应的数据，聚合后，再返回给用户 |
|  | mail-provider | 4000 |  |  |  |  |  |  |

# 校验服务,这里假定服务开启了6060的http监听端口。检验结果为ok表明服务正常启动

curl -s "http://127.0.0.1:6060/health"

# 绘图数据的数据链路是：agent->transfer->graph->query->dashboard

# 报警数据的数据链路是：agent->transfer->judge

# 如果上报的数据不带tags，访问方式是这样的:

curl http://127.0.0.1:6071/history/host01/agent.alive

#cfg.json中的各配置项，可以参考 https://github.com/open-falcon/transfer/blob/master/README.md

合并portal,links,uic



## 系统部署

批量删除falcon进程

kill -9 `ps -ef|grep falcon|awk '{print $2}'`

kill -9 `ps -ef|grep dashboard|awk '{print $2}'`

kill -9 `ps -ef|grep portal|awk '{print $2}'`

kill -9 `ps -ef|grep "falcol\|portal\|dashboard"|awk '{print $2}'`

docker pull index.alauda.cn/tutum/centos

docker run -d -p 0.0.0.0:2222:22 –p 8081:8081 -e ROOT\_PASS="mypass" tutum/centos

docker run -d --name tutum -p 0.0.0.0:2222:22 -p 8081:8081 -e ROOT\_PASS="mypass" tutum/centos

yum install -y mariadb-server mariadb redis gcc mysql-devel git

docker run --name tutum --privileged -ti -e "container=docker" -v /sys/fs/cgroup:/sys/fs/cgroup -d -p 0.0.0.0:2222:22 -p 8081:8081 -e ROOT\_PASS="mypass" tutum/centos /usr/sbin/init

docker run --name tutum2 --privileged -ti -e "container=docker" -v /sys/fs/cgroup:/sys/fs/cgroup -d -p 0.0.0.0:2222:22 -p 8081:8081 -e ROOT\_PASS="mypass" tutum/centos /usr/sbin/init

docker cp python-lib tutum:/home/work/open-falcon

yum -y install mariadb-server mariadb

MariaDB An enhanced, drop-in replacement for MySQL server. RHEL/CentOS v7.x shifts from MySQL to MariaDB for its database management system needs. Type the following yum command to install MariaDB server:

sudo yum install mariadb-server mariadb

To start mariadb, type:

sudo systemctl start mariadb.service

To make sure the mariadb service start automatically at the boot time, enter:

sudo systemctl enable mariadb.service

docker cp of-release-v0.1.0.tar.gz tutum:/home/work/open-falcon

docker cp python-lib tutum:/home/work/open-falcon/dashboard

docker cp python-lib tutum:/home/work/open-falcon

https://pypi.python.org/pypi/MySQL-python/1.2.5

<https://pypi.python.org/pypi/Flask/0.11>

https://pypi.python.org/pypi/MarkupSafe

<https://pypi.python.org/pypi/Flask/0.10.1>

https://pypi.python.org/pypi/gunicorn/19.3.0

./env/bin/pip install ../python-lib/itsdangerous-0.24.tar.gz

./env/bin/pip install ../python-lib/MarkupSafe-0.23.tar.gz

./env/bin/pip install ../python-lib/Jinja2-2.7.2.tar.gz

./env/bin/pip install ../python-lib/Werkzeug-0.9.4.tar.gz

./env/bin/pip install ../python-lib/Werkzeug-0.11.10.tar.gz

./env/bin/pip install ../python-lib/gunicorn-18.0.tar.gz

./env/bin/pip install ../python-lib/python-dateutil-2.2.tar.gz

./env/bin/pip install ../python-lib/requests-2.3.0.tar.gz

./env/bin/pip install ../python-lib/MySQL-python-1.2.5.zip

./env/bin/pip install ../python-lib/gunicorn-19.3.0.tar.gz

./env/bin/pip install ../python-lib/Flask-0.10.1.tar.gz

### 测试环境

https://test.yihecloud.com/monitor/

192.168.1.81部署

数据库192.168.10.82

Cd /opt/ /opt/tomcat-monitor

upgrade.sh

### 开发环境部署

192.168.31.220

Root:123456

/opt/monitor

### 添加systemd服务

echo "[Unit]

Description=flacon hbs initialization

[Service]

WorkingDirectory=/opt/open-falcon/hbs

ExecStart=/bin/bash -c './falcon-hbs -c cfg.json'

Restart=on-failure

[Install]

WantedBy=multi-user.target

" > /lib/systemd/system/falcon-hbs.service

systemctl daemon-reload  
 systemctl start falcon-hbs;systemctl status falcon-hbs

### Open-Falcon部署 - 简书

http://www.jianshu.com/p/a5fcd5c048f1

本文并不分析Open-Falcon的架构或者选用它的原因，官方的文档在[这里](http://book.open-falcon.com/zh/intro/README.html)，虽然还不够完善。不过这也是我写这篇的原因，官方文档并没能把整个部署过程连在一起，而且个别地方有点问题。我在这篇文章中就不介绍各个组件的作用和功能了，只是单纯的介绍如何从零部署。

安装

下载

wget https://github.com/XiaoMi/open-falcon/releases/download/0.0.5/open-falcon-0.0.5.tar.gz -O open-falcon.tar.gz

解压

mkdir tmp

tar -zxvf open-falcon.tar.gz -C ./tmp

基础环境

sudo apt-get install redis-server

sudo apt-get install mysql-server

pip install virtualenv

#数据库初始化的代码来源于官方文档

git clone https://github.com/open-falcon/scripts.git

cd scripts

mysql -h localhost -u root -p < db\_schema/graph-db-schema.sql

mysql -h localhost -u root -p < db\_schema/dashboard-db-schema.sql

mysql -h localhost -u root -p < db\_schema/portal-db-schema.sql

mysql -h localhost -u root -p < db\_schema/links-db-schema.sql

mysql -h localhost -u root -p < db\_schema/uic-db-schema.sql

配置

数据库连接的配置格式是: username:password@tcp(path:port)/xxxx

agent

mv cfg.example.json cfg.json && ./control start && ./control tail  
开始监听1988端口,查看log.  
默认端口是1988, 可以打开 <http://127.0.0.1:1988> 查看一个比较简单的web dashboard. 没什么特殊需要的话, 可以使用默认配置.

hbs

mv cfg.example.json cfg.json && ./control start && ./control tail  
心跳服务默认http端口是6030, rpc 端口 6031

transfer

mv cfg.example.json cfg.json && ./control start && ./control tail  
默认http端口是6060, rpc端口8433

judge

mv cfg.example.json cfg.json && ./control start && ./control tail  
http端口6081, rpc端口 6080  
配置项中注意alarm的 redis链接 和hbs的server地址,如果修改过请记得对应.

graph

mv cfg.example.json cfg.json && ./control start && ./control tail  
rpc端口 6070, http端口 6071  
数据库文件存储在/home/work/data/6070 启动报错的话, 换sudo 或者 root 用户启动.  
注意修改数据库连接.

上面的服务配置完成数据就开始采集了.

dashboard

virtualenv env

source env/bin/activate

./env/bin/pip install -r pip\_requirements.txt

# 使用

./env/bin/python wsgi.py

# 或者用

deactivate && ./control start && ./control tail

可以更改为自定义端口.这里可以查看Endpoints 的相关数据并绘图.

query

mv cfg.example.json cfg.json && ./control start && ./control tail  
只要修改cfg.json 文件即可, 注意还有 graph\_backends.txt 文件

fe

mv cfg.example.json cfg.json && ./control start && ./control tail  
注意配置项目中的数据库连接, 以及下面的shortcut 中需要配置外网可以访问的地址,如果不是在服务器部署的话,这里默认也没有关系.

portal

这里要注意一个坑. 如果使用./control start 启动服务, 并且更改了默认端口的, 请配置 gunicorn.conf 中的bind项. 如果是使用python wsgi.py启动的话, 修改wsgi.py中的端口即可.  
同时在配置 frame/config.py 的时候, 要注意所谓UIC\_ADDRESS 选项的配置, 其实就是填写上面fe 模块的地址.

UIC\_ADDRESS = {

'internal': 'http://127.0.0.1:port', #你的内网地址

'external': 'http://your\_fe\_name', #外网访问的地址, 如果是本地部署,这里可以和内网地址一样

}

这里确实想对web界面的操作吐槽一下,相当不人性化.而且配置告警的时候,真的没有把整个流程联系在一起啊喂!这里的配置可以创建一个template 然后再创建监控主机,进行绑定.  
在模板中配置的callback() 非常好用.可以在这里直接写告警消息的推送接口,使用第三方服务也ok.

alarm

mv cfg.example.json cfg.json && ./control start && ./control tail  
需要配置的就是自定义的端口监控地址, 和下面的api, 其他部分保持默认即可.  
在[它的文档](http://book.open-falcon.com/zh/install_from_src/alarm.html)中写着下面这样的说明

{...

"api": {

"portal": "http://falcon.example.com", # 内网可访问的portal的地址

"uic": "http://uic.example.com", # 内网可访问的uic(或fe)的地址

"links": "http://link.example.com" # 外网可访问的links的地址

}

}

但是如果按照这样配的话,所有东西就只能在内网访问了.这明显是不符合我们的期望的. 所以这里要全部配置成外网可以访问的地址,防止踩坑.

sender

这个组件用于调用自己提供的短信和邮件接口, 编辑 cfg.json 添加

{...

'api': {

'sms': 'http://your\_send\_sms\_api',

'mail': 'http://your\_send\_mail\_api'

}

}

在调用的时候, 会把消息先写入redis队列中, 之后再调用接口,进行消息发送.  
worker 参数是对队列的配置.

links

mv cfg.example.json cfg.json && ./control start && ./control tail  
告警合并组件, python应用, 记得修改默认端口即可.

其他

task

监控自检程序

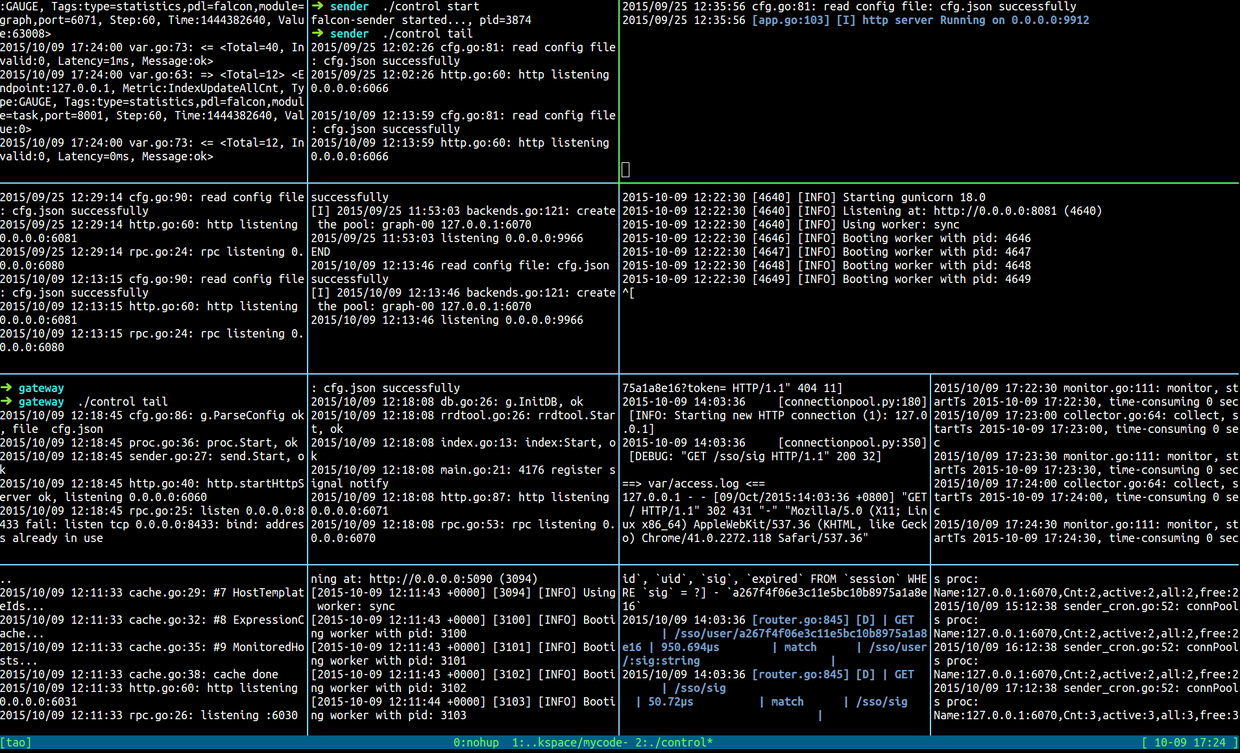
gateway

没有跨机房问题可以忽略.(hhah

redis-monitor.py

scripts 文件夹中一个很简单的脚本, 把redis info 中的数据读出,写入数据库, 供监控使用.

最后附图一张，前几天我启动这个服务时候的终端。



2015-10-09-172438\_1911x1160\_scrot.png

文／TaoBeier（简书作者）  
原文链接：http://www.jianshu.com/p/a5fcd5c048f1  
著作权归作者所有，转载请联系作者获得授权，并标注“简书作者”。

## 需求

### 告警常用设置

http://book.open-falcon.org/zh/faq/linux-metrics.html

all(#3): 最新的3个点都满足阈值条件则报警

max(#3): 对于最新的3个点，其最大值满足阈值条件则报警

min(#3): 对于最新的3个点，其最小值满足阈值条件则报警

sum(#3): 对于最新的3个点，其和满足阈值条件则报警

avg(#3): 对于最新的3个点，其平均值满足阈值条件则报警

diff(#3): 拿最新push上来的点（被减数），与历史最新的3个点（3个减数）相减，得到3个差，只要有一个差满足阈值条件则报警

pdiff(#3): 拿最新push上来的点，与历史最新的3个点相减，得到3个差，再将3个差值分别除以减数，得到3个商值，

mem.memused.percent

cpu.busy

告警函数 采用： avg(#3): 对于最新的3个点，其平均值满足阈值条件则报警

## common

### model包event.go

**func** (this \*Event) String() string {

**return** fmt.Sprintf(

        "<Endpoint:%s, Status:%s, Strategy:%v, Expression:%v, LeftValue:%s, CurrentStep:%d, PushedTags:%v, TS:%s>",

        this.Endpoint,

        this.Status,

        this.Strategy,

        this.Expression,

        utils.ReadableFloat(this.LeftValue),

        this.CurrentStep,

        this.PushedTags,

        this.FormattedTime(),

    )

}

### model.Strategy

## dashboard

pip install <path to archive>

https://pypi.python.org/pypi/Flask/0.11

https://pypi.python.org/pypi/Flask/0.10.1

docker cp ./python-lib jolly\_austin:/home/work/open-falcon/dashboard

#制作镜像

docker build -t mydashboard .

docker run -d -it mydashboard bash

docker run -d -it mydashboard python wsgi.py

docker run -d --name mydashboard --restart=always -p 8081:8081 mydashboard python wsgi.py

### 显示均值，极值

E:\workspace\open-falcon\dashboard\rrd\static\js\jquery.flot.js

fragments.push(**'<th>&nbsp;</th><th>last</th><th>min</th><th>avg</th><th>max</th>'**);

### alpine docker file

FROM alpine

MAINTAINER liyang <liyang@yihecloud.com>

WORKDIR /opt

WORKDIR /opt

RUN apk update && apk add bash

RUN apk add python

RUN apk add py-mysqldb

RUN apk add py-pip

RUN apk add py-virtualenv

RUN apk add openssl

#RUN apk add mysql

RUN virtualenv ./env

ADD . /opt/

RUN pip install ./python-lib/itsdangerous-0.24.tar.gz

RUN pip install ./python-lib/MarkupSafe-0.23.tar.gz

RUN pip install ./python-lib/Jinja2-2.7.2.tar.gz

RUN pip install ./python-lib/Werkzeug-0.9.4.tar.gz

RUN pip install ./python-lib/Werkzeug-0.11.10.tar.gz

RUN pip install ./python-lib/gunicorn-18.0.tar.gz

RUN pip install ./python-lib/six-1.10.0.tar.gz

RUN pip install ./python-lib/python-dateutil-2.2.tar.gz

RUN pip install ./python-lib/requests-2.3.0.tar.gz

#RUN pip install ./python-lib/MySQL-python-1.2.5.zip

RUN pip install ./python-lib/gunicorn-19.3.0.tar.gz

RUN pip install ./python-lib/Flask-0.10.1.tar.gz

expose 8081

CMD /bin/sh python wsgi.py

### api/counters

<http://192.168.1.135:8081/api/counters>

E:\workspace\open-falcon\dashboard\rrd\model\endpoint\_counter.py

search\_in\_endpoint\_ids

graph数据库中

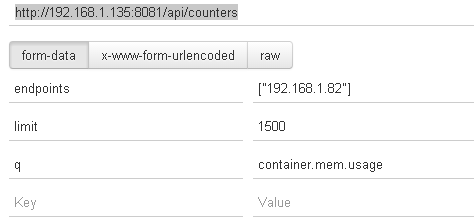
select id, endpoint\_id, counter, step, type from endpoint\_counter where endpoint\_id

endpoints:

["192.168.1.55"]

q:

limit:50



## Hbs

### @更新host信息open-falcon\hbs\db\agent.go

E:\workspace\yh\OpenBridge-passos-proxy\open-faclon\src\github.com\open-falcon\hbs\db\agent.go

"insert into host(hostname, ip, agent\_version, plugin\_version, cpu\_num, mem\_num) values ('%s', '%s', '%s', '%s','%s','%s') on duplicate key update ip='%s', agent\_version='%s', plugin\_version='%s',cpu\_num='%s', mem\_num='%s'",

### open-falcon\hbs\db\agent.go每20分钟清理docker

E:\workspace\yh\OpenBridge-passos-proxy\open-faclon\src\github.com\open-falcon\hbs\db\agent.go

select timestampdiff(MINUTE,update\_time,now()) , docker.\* from docker ;

**func** CleanDocker() {  
 // sync, just delete  
 sql := fmt.Sprintln("delete from docker where timestampdiff(MINUTE,update\_time,now()) > 20 ")  
 \_, err := DB.Exec(sql)  
 **if** err != nil {  
 log.Println("exec", sql, "fail", err)  
 }  
}

### open-falcon\hbs\rpc\agent.go agent调用hbs的rpc方法ReportStatus

E:\workspace\yh\OpenBridge-passos-proxy\open-faclon\src\github.com\

open-falcon\hbs\rpc\agent.go

### Cache包cache.go

定时更新缓存

func LoopInit() {

    for {

        time.Sleep(time.Minute)

        GroupPlugins.Init()

        GroupTemplates.Init()

        HostGroupsMap.Init()

        HostMap.Init()

        TemplateCache.Init()

        Strategies.Init(TemplateCache.GetMap())

        HostTemplateIds.Init()

        ExpressionCache.Init()

        MonitoredHosts.Init()

    }

}

### Cache包strategies.go

根据hostname获取id

    hid, exists := HostMap.GetID(hostname)

根据hid获取gids

gids, exists := HostGroupsMap.GetGroupIds(hid)

根据gid获取tids

tids, exists := GroupTemplates.GetTemplateIds(gid)

### Db包host.go

QueryHosts

获取所有主机

**func** QueryHosts() (**map**[string]int, error)

    sql := "select id, hostname from host"

### Db包strategy.go

QueryStrategies

**func** QueryBuiltinMetrics(tids string) ([]\*model.BuiltinMetric, error) {

### GetStrategies在rpc包中

## No-data

### open-falcon/nodata: 一段时间内数据上报中断则触发告警

https://github.com/open-falcon/nodata

nodata用于检测监控数据的上报异常。nodata和实时报警judge模块协同工作，过程为: 配置了nodata的采集项超时未上报数据，nodata生成一条默认的模拟数据；用户配置相应的报警策略，收到mock数据就产生报警。采集项上报异常检测，作为judge模块的一个必要补充，能够使judge的实时报警功能更加可靠、完善。

## Rrdtools

### 指定resolution

[root@dns1 bob]# rrdtool graph 5.png   
> -S 1200 \                                # -S 指定 resolution=1200  
> --start now-120000 \  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE:step=300 \        # :step 指定 resolution=300  
> AREA:value1#ff0000       
481x154  
[root@dns1 bob]#

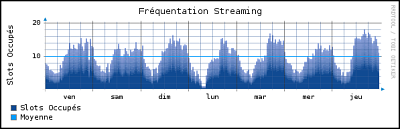
### RRDtool的用法（结合实例） - 推酷

http://www.tuicool.com/articles/JJZVNb

RRDtool是指Round Robin Database工具，即环状数据库。从功能上说，RRDtool可用于数据存储+数据展示。著名的网络流量绘图软件MRTG和集群监控系统Ganglia都使用的RRDtool。

数据存储方面，RRDtool采用“Round Robin”模式存储数据。所谓“Round Robin”是一种存储数据的方式，使用固定大小的空间来存储数据，并有指针指向最新的数据的位置。我们可以把用于存储数据的数据库空间看成一个圆，上面有很多刻度，这些刻度所在的位置就代表用于存储数据的地方。所谓指针，可以认为是从圆心指向这些刻度的一条线。指针会随着数据的读写自动移动。要注意的是，这个圆没有起点和终点，所以指针可以一直移动，而不担心到达终点后无法继续写入的问题。在一段时间后，当所有的空间都存满数据，就又从头开始存放。这样整个存储空间的大小就是一个固定的数值。RRDtool所使用数据库文件的后缀名是“.rrd”。

数据展示方面，RRDtool可以看作是一个强大的绘图引擎。下图是其官网上的一张效果图，我们大致了解RRDtool的绘图能力。



RRDtool官网链接： [*http://oss.oetiker.ch/rrdtool/*](http://oss.oetiker.ch/rrdtool/)

RRDtool的使用分为三个部分，建库、更新数据、绘图（具体使用shell命令）。因此，我们也分上述三个部分介绍基本语法。

为表述清晰，这里将建库命令写成分段形式，实际应用时应写成一串。

rrdtool create cpu.rrd *# 数据库名称*

--start $(date -d '1 days ago' +%s) *# 开始时间*

--step 15 *# 更新数据时间间隔*

DS:cpu\_user:GAUGE:120:0:NaN *# DS:cpu\_user,相当于变量名;后面的是DST：GAUGE，相当于数据类型;后面120是heartbeat，是最大没有数据的间隔;后面两个NaN分别是最小值、最大值限制。*

DS:cpu\_system:GAUGE:120:0:NaN

DS:cpu\_wio:GAUGE:120:0:NaN

DS:cpu\_idle:GAUGE:120:0:NaN

RRA:AVERAGE:0.5:1:244 *# RRA是数据存储的形式，数据表*

RRA:AVERAGE:0.5:24:244 *# CF合并统计 有average、max、min、last四种*

RRA:AVERAGE:0.5:168:244 *# 0.5是xff，表示缺少数据量比例大于0.5时，数据显示为空。*

RRA:AVERAGE:0.5:672:244 *# PDP，计算出来的一个数据点，如平均值等*

RRA:AVERAGE:0.5:5760:374 *# CDP，使用多个PDP合并成一个CDP，CDP是真正存入RRA的值，也是绘图时使用的值，1、24、168、672等表示多少个PDP合并成一个CDP*

具体参数意义大家参见注释，本段代码的大意是创建一个rrd数据库cpu.rrd，保存cpu相关信息，每15秒更新一次数据。

更新比较简单，就是定时向数据库（即.rrd文件）中写入数据。每次写入命令，类似下面指令。

rrdtool updatev /**var**/lib/monitor/rrds/server/cpu.rrd 1382646278:0.733211:0.433261:1.516414:97.317114

**# /var/lib/monitor/rrds/server/cpu.rrd 是数据库文件**

**# 1382646278是时间戳**

**# 0.733211:0.433261:1.516414:97.317114是写入的具体数值，分别指代cpu\_user、cpu\_system、cpu\_wio、cpu\_idle。**

实际使用时应当写一个程序定时获取cpu利用率，并执行上述命令，将数据写入数据库。

与建库时类似，为表述清晰，这里将绘图命令写成分段形式，实际应用时应写成一串。

/usr/bin/rrdtool graph /home/xx/cpu.png

--start '-3600' --end N *// 过去一小时的时间*

--width 385 --height 190 *// 图片大小*

--title '过去一小时CPU使用情况' --upper-limit 100 --lower-limit 0 *// 题目和上下限*

--vertical-label 百分比 --rigid

DEF:'cpu\_user'='/var/lib/monitor/rrds/server/cpu.rrd':'cpu\_user':AVERAGE **# 获得变量cpu\_user**

AREA:'cpu\_user'**#FF0000:'用户' // 图形形式，包括AREA、STACK、LINE等**

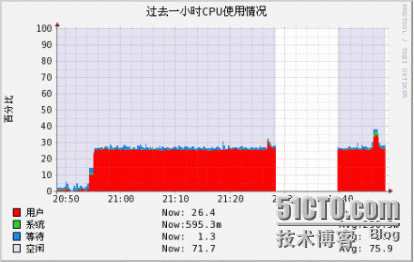
VDEF:cpu\_user\_last=cpu\_user,LAST *// 变量定义，取具体值*

VDEF:cpu\_user\_avg=cpu\_user,AVERAGE

GPRINT:'cpu\_user\_last':' Now\:%5.1lf%s' *// 在图片中打印数值*

GPRINT:'cpu\_user\_avg':' Avg\:%5.1lf%s\j' DEF:'cpu\_system'='/var/lib/monitor/rrds/server/cpu.rrd':'cpu\_system':AVERAGE STACK:'cpu\_system'**#33cc33:'系统' VDEF:cpu\_system\_last=cpu\_system,LAST VDEF:cpu\_system\_avg=cpu\_system,AVERAGE GPRINT:'cpu\_system\_last':' Now\:%5.1lf%s' GPRINT:'cpu\_system\_avg':' Avg\:%5.1lf%s\j' DEF:'cpu\_wio'='/var/lib/monitor/rrds/server/cpu.rrd':'cpu\_wio':AVERAGE STACK:'cpu\_wio'#1C86EE:'等待' VDEF:cpu\_wio\_last=cpu\_wio,LAST VDEF:cpu\_wio\_avg=cpu\_wio,AVERAGE GPRINT:'cpu\_wio\_last':' Now\:%5.1lf%s' GPRINT:'cpu\_wio\_avg':' Avg\:%5.1lf%s\j' DEF:'cpu\_idle'='/var/lib/monitor/rrds/server/cpu.rrd':'cpu\_idle':AVERAGE STACK:'cpu\_idle'#e2e2f2:'空闲' VDEF:cpu\_idle\_last=cpu\_idle,LAST VDEF:cpu\_idle\_avg=cpu\_idle,AVERAGE GPRINT:'cpu\_idle\_last':' Now\:%5.1lf%s' GPRINT:'cpu\_idle\_avg':' Avg\:%5.1lf%s\j'**

得到的图片如下。我刻意停止更新数据一段时间，绘图得到的图片也在相应时间段内监控显示没有数据。



[*RRDTOOL 百度文库*](http://wenku.baidu.com/link?url=bp_KkCQLduld7Ul88v9D8XRfBcniSeYVcwoZpq9CjwMh7GmxOrAoaM_E9IXm9p86Gu1NEenV7bnZl57C27D3dBt2QvzxsH-sQM6qsWztrKy)

[*RRDtool绘图使用详细*](http://www.360doc.com/content/07/0302/20/15540_382048.shtml)

### RRDtool绘图使用详细

http://www.360doc.com/content/07/0302/20/15540\_382048.shtml

**RRDtool 的定义**  
  
   RRDtool 代表 “Round Robin Database tool” ，作者同时也是 MRTG 软件的发明人。官方站点位于http://oss.oetiker.ch/rrdtool/ 。   
  
    所谓的“Round Robin” 其实是一种存储数据的方式，使用固定大小的空间来存储数据，并有一个指针指向最新的数据的位置。我们可以把用于存储数 据的数据库的空间看成一个圆，上面有很多刻度。这些刻度所在的位置就代表用于存储数据的地方。所谓指针，可以认为是从圆心指向这些刻度的一条直线。指针会 随着数据的读写自动移动。要注意的是，这个圆没有起点和终点，所以指针可以一直移动，而不用担心到达终点后就无法前进的问题。在一段时间后，当所有的空间 都存满了数据，就又从头开始存放。这样整个存储空间的大小就是一个固定的数值。所以RRDtool 就是使用类似的方式来存放数据的工具， RRDtool 所使用的数据库文件的后缀名是‘.rrd’。

**RRDtool 的特殊之处**

A） 首先 RRDtool 存储数据，扮演了一个后台工具的角色。但同时 RRDtool 又允许创建图表，这使得RRDtool看起来又像是前端工具。其他的数据库只能存储数据，不能创建图表。   
  
B） RDtool 的每个 rrd 文件的大小是固定的，而普通的数据库文件的大小是随着时间而增加的。  
  
C） 其他数据库只是被动的接受数据， RRDtool 可以对收到的数据进行计算，例如前后两个数据的变化程度（rate of  change），并存储该结果。   
  
D） RRDtool 要求定时获取数据，其他数据库则没有该要求。如果在一个时间间隔内（heartbeat）没有收到值，则会用 UNKN 代替，其他数据库则不会这样。  
  
**建立 RRD 数据库**

建库实际上就是建立后缀名为 .rrd 的 RRD 文件。  
  
**一）语法格式**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864866)

rrdtool create filename [--start|-b start time] [--step|-s step]   
                         [DS:ds-name:DST:dst arguments]    
                         [RRA:CF:cf arguments]

其中 filename 、DS 部分和 RRA 部分是必须的。其他两个参数可免。  
  
**二）参数解释**  
  
A）**<filename> ：**默认是以 .rrd 结尾，但也以随你设定。  
  
B） **--step ：**就是 RRDtool “期望” 每隔多长时间就收到一个值。和 MRTG 的 interval 同样含义。默认是5分钟。我们的脚本也应该是  
  
          每5分钟运行一次。  
  
C） **--start ：**给 出 RRDtool 的第一个记录的起始时间。RRDtool 不会接受任何采样时间小于或者等于指定时间的数据。也就是说 –-start指定了数据库最早的那个记录是从什么时候开始的。如果 update 操作中给出的时间在 –-start 之前，则 RRDtool拒绝接受。--satrt 选项也是可选的。按照 我们在前一篇中的设定，则默认是当前时间减去 600\*300秒，也就是50个小时前。 如果你想指定--start 为1天前，可以用

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864866)

--start $(date -d ‘1 days ago‘ +%s)

注意，--start 选项的值必须是 timestamp 的格式。  
  
D） **DS ：**DS 用于定义 Data Soure 。也就是用于存放脚本的结果的变量名(DSN)。  
  
      就是我们前面提到的 eth0\_in ,eth0\_out, lo\_in , lo\_out 。DSN 从 1-19 个字符，必须是 0-9,a-z,A-Z 。  
  
E） **DST ：**DST 就是 Data Source Type 的意思。有 COUNTER、GUAGE、DERIVE、ABSOLUTE、COMPUTE 5种。由于网卡流量属于计数器型，所以这里应该为 COUNTER 。  
  
F） **RRA ：**RRA 用于指定数据如何存放。我们可以把一个RRA 看成一个表，各保存不同 interval 的统计结果  
  
G）**PDP ：**Primary Data Point 。正常情况下每个 interval RRDtool 都会收到一个值；RRDtool 在收到脚本给来的值后 会计算出另外一个值（例如平均值），这个 值就是 PDP ；这个值代表的一般是“xxx/秒”的含义。注意，该值不一定等于RRDtool  收到的那个值。除非是GAUGE ，可以看下面的例子就知道了        
  
H） **CF ：**CF 就是 Consolidation Function 的缩写。也就是合并（统计）功能。有 AVERAGE、MAX、MIN、LAST 四种分别表示对多个PDP 进行取平均、取最大值、取最小值、取当前值四种类型。具体作用等到 update 操作时 再说。  
  
I） CDP ：Consolidation Data Point 。RRDtool 使用多个 PDP 合并为（计算出）一个 CDP。也就是执行上面 的CF 操作后的结果。这个值就是存入 RRA的数据，绘图时使用的也是这些数据。  
  
  
**三）再说 DST**   
  
    DST 的选择是十分重要的，如果选错了 DST ，即使你的脚本取的数据是对的，放入 RRDtool 后也是错误的，更不用提画出来的图是否有意义了。  
      
    如何选择 DST 看下面的描述 ：  
  
   A）COUNTER ：必须是递增的，除非是计数器溢出（overflows）。在这种情况下，RRDtool 会自动修改收到的值。例如网络接口流量、收到的packets 数量都属于这一类型。  
   B）DERIVE：和 COUNTER 类似。但可以是递增，也可以递减，或者一会增加一会儿减少。  
   C）ABSOLUTE ：ABSOLUTE 比较特殊，它每次都假定前一个interval的值是0，再计算平均值。  
   D）GAUGE ：GAGUE 和上面三种不同，它没有“平均”的概念，RRDtool 收到值之后字节存入 RRA 中  
   E）COMPUTE ：COMPUTE 比较特殊，它并不接受输入，它的定义是一个表达式，能够引用其他DS并自动计算出某个值。例如

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864866)

DS:eth0\_bytes:COUNTER:600:0:U DS:eth0\_bits:COMPUTE:bytes,8,\*

则 eth0\_bytes 每得到一个值，eth0\_bits 会自动计算出它的值：将 eth0\_bytes 的值乘以 8 。不过 COMPUTE 型的 DS 有个限制，只能应用它所在的 RRD 的 DS ，不能引用其他 RRD 的 DS。 COMPUTE 型 DS 是新版本的 RRDtool 才有的，你也可以用 CDEF 来实现该功能。  
F）AVERAGE 类型适合于看“平均”情况，例如一天的平均流量，。所以 AVERAGE 适用于需要知道 ‘xxx/秒’ 这样的需求。但采用 AVERAGE 型时，并不知道 在每个 CDP 中（假设30分钟平均，6个PDP组成）之中，流量具体是如何变化的，什么时候高，什么时候低。这于需要用到别的统计类型了  
G）MAXIMUM 、MINIMUM不适用想知道“xxx/秒”这样的需求，而是适用于想知道某个对象在各个不同时刻的表现的需求，也就是着重点在于各个时间点。  
        例如要看某个接口在一天内有没有超过50Mb 流量的时候就要用 MAXIMUM  
       例如要看磁盘空间的空闲率在一天内有没有低于 20% 的时候就要用 MINIMUM  
H） LAST 类型适用于 “累计”的概念，例如从xxx时候到目前共累计xxxx 这样的需求。例如邮件数量，可以用 LAST 来表示 30 分钟内总共收到多少个邮件，同样 LAST 也没有平均的概念，也就是说不适用于 ‘xxx/秒’ 这样的需求，例如你不能说平均每秒钟多少封邮件这样的说法；同样也不适用于看每个周期内的变化，例如30分钟内共收到100封邮件，分别是 ：第一个5分钟20封，第二个5分钟30封，第三个5分钟没有，第4个5分钟10封，第5个5分钟也没有，第6个5分钟40封。如果用 MAXIMUM 或者 MINIMUM 就不知道在30分钟内共收到100封邮件，而是得出30和0。所以 LAST 适用于每隔一段时间被观察 对象就会复位的情况。例如每30分钟就收一次邮件，邮件数量就是 LAST 值，同时现有的新邮件数量就被清零；到下一个30分钟再收一次邮件，又得到一个 30  分钟的 LAST 值。          
         这样就可以得得出“距离上一次操作后到目前为止共xxx”的需求。（例如距离上一次收取邮件后又共收到100封新邮件）  
  
**四）DST 实例说明**  
  
这样说可能还是比较模糊，可以看下面的例子，体会一下什么是 DST 和 PDP  ：

**QUOTE:**

Values = 300, 600, 900, 1200        # 假设 RRDtool 收到4个值，分别是300，600，900，1200  
  
Step = 300 seconds                    # step 为 300  
  
COUNTER = 1,1, 1,1                         # （300-0）/300，（600-300）/300，（900-600）/300，（1200-900）/300 ，所以结果为 1，1，1，1  
  
DERIVE = 1,1,1,1                         # 同上  
  
ABSOLUTE = 1,2,3,4                   # (300-0)/300,(600-0)/300 , (900-0)/300, (1200-0)/300，所以结果为 1，2，3，4  
  
GAUGE = 300,600,900,1200          # 300 , 600 ,900 ,1200 不做运算，直接存入数据库

所以第一行的 values 并不是 PDP ，后面4行才是 PDP   
  
  
**五）开始建库**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864866)

[root@dns1 root]# rrdtool create eth0.rrd \  
> --start $(date –d ‘1 days ago’ +%s) \  
> --step 300 \  
> DS:eth0\_in:COUNTER:600:0:12500000 \        #  600 是 heartbeat；0 是最小值；12500000 表示最大值；   
> DS:eth0\_out:COUNER:600:0:12500000 \        # 如果没有最小值/最大值，可以用 U 代替，例如 U:U  
> RRA:AVERAGE:0.5:1:600 \        # 1 表示对1个 PDP 取平均。实际上就等于 PDP 的值  
> RRA:AVERAGE:0.5:4:600 \        # 4 表示每4个 PDP 合成为一个 CDP，也就是20分钟。方法是对4个PDP取平均，   
> RRA:AVERAGE:0.5:24:600 \  # 同上，但改为24个，也就是24\*5=120分钟=2小时。  
> RRA:AVERAGE:0.5:288:730        # 同上，但改为288个，也就是 288\*5=1440分钟=1天  
[root@dns1 root]#

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864866)

root@dns1 bob]# ll -h eth0.rrd  
-rw-r--r--    1 root     root          41K 11月 19 23:16 eth0.rrd  
[root@dns1 bob]#

有的人可能会问，上面有两个 DS，那 RRA 中究竟存的是那个 DS 的数据呢？实际上，这些 RRA 是共用的，你只需建立一个 RRA，它就可以用于全部的 DS 。  
所以在定义 RRA 时不需要指定是给那个 DS 用的。  
  
  
**六）什么是 CF**   
  
以第2个RRA 和 4，2，1，3 这4个 PDP 为例  
  
**AVERAGE** ：则结果为 (4+2+1+3)/4=2.5  
  
**MAX** ：结果为4个数中的最大值 4  
  
**MIN** ：结果为4个数中的最小值1  
  
**LAST** ：结果为4个数中的最后一个 3  
  
同理，第三个RRA和第4个RRA则是每24个 PDP、每288个 PDP 合成为1个 CDP  
  
**七）解释度（Resolution）**  
  
这里要提到一个 Resolution 的概念，在官方文档中多处提到 resolution 一词。Resolution 究竟是什么？Resolutino 有什么用？  
  
举个例子，如果我们要绘制1小时的数据，也就是60分钟，那么我们可以从第一个RRA 中取出12个 CDP 来绘图；也可以从第2个 RRA中取出2个 CDP 来绘图。到底 RRDtool 会使用那个呢？  
  
让我们看一下 RRA 的定义 ：RRA:AVERAGE:0.5:4:600 。  
  
Resolution 就等于 4 \* step = 4 \* 300 = 1200 ，也就是说 ，resolution 是每个CDP 所代表的时间范围，或者说 RRA 中每个 CDP（记录）之间的时间间隔。所以第一个 RRA 的 resolution 是 1\* step=300，第2是 1200，第三个是 24\*300=7200，第4个 RRA 是 86400 。  
  
默认情况下，RRDtool 会自动挑选合适的 resolution 的那个 RRA 的数据来绘图。我们大可不必关心它。但如果自己想取特定 RRA 的数据，就需要用到它了。  
  
关于 Resolution 我们还会在 fetch 和 graph 中提到它。  
  
  
**八）xff 字段**  
  
细心的朋友可能会发现，在 RRA 的定义中有一个数值，固定是 0.5 ，这个到底是什么东东呢？  
  
这个称为 xff 字段，是 **xfile factor** 的缩写。让我们来看它的定义 ：

**QUOTE:**

The xfiles factor defines what part of a consolidation interval may be made up from \*UNKNOWN\* data while  
  
the consolidated value is still regarded as known. It is given as the ratio of allowed \*UNKNOWN\* PDPs to   
  
the number of PDPs in the interval. Thus, it ranges from 0 to 1 (exclusive)

这个看起来有点头晕，我们举个简单的例子 ：例如

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864866)

RRA:AVERAGE:0.5:24:600

这个 RRA 中，每24个 PDP （共两小时）就合成为一个 CDP，如果这 24 个 PDP 中有部分值是 UNKNOWN （原因可以很多），例如1个，那么这个 CDP   
  
合成的结果是否就为 UNKNOWN 呢？  
  
不是的，这要看 xff 字段而定。Xff 字段实际就是一个比例值。0.5 表示一个 CDP 中的所有 PDP 如果超过一半的值为 UNKNOWN ，则该 CDP 的值就被标为  
  
UNKNOWN。也就是说，如果24个 PDP中有12个或者超过12个 PDP 的值是 UNKNOWN ，则该 CPD 就无法合成，或者合成的结果为 UNKNOWN；  
  
如果是11个 PDP 的值为 UNKNOWN ，则该 CDP 的值等于剩下 13  个 PDP 的平均值。  
  
如果一个 CDP 是有2个 PDP 组成，xff 为 0.5 ，那么只要有一个 PDP 为 UNKNOWN ，则该 PDP 所对应的 CDP 的值就是 UNKNOWN 了 

**使用RRDtool 进行绘图**

**一）前言**使用RRDtool 我们最关心什么？当然是把数据画出来了。虽然前面谈了很多，但这些都是基础来的。掌握好了，可以让你在绘图时更加得心应手。  
  
本来还有 RPN （反向波兰表达式）一节的，但考虑一下，觉得还是放到后面，先从基本的绘图讲起。  
  
这一节的内容虽然很多，但基本都是实验性的内容，只要多试几次就可以了。  
  
**二、graph 语法**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

rrdtool graph filename [option ...]   
   [data definition ...]  
   [data calculation ...]          
   [variable definition ...]  
   [graph element ...]  
   [print element ...]

其中的 data definiton、variable definition 、data calculation、分别是下面的格式

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

DEF:<vname>=<rrdfile>:<ds-name>:<CF>[:step=<step>][:start=<time>][:end=<time>][:reduce=<CF>]  
VDEF:vname=RPN expression  
CDEF:vname=RPN expression

其中 filename 就是你想要生成的图片文件的名称，默认是 png 。你可以通过选项修改图片的类型，可以有 PNG、SVG、EPS、PDF四种。  
  
A）DEF 是 Definition （定义）的意思。定义什么呢？你要绘图，总要有数据源吧？DEF 就是告诉 RRDtool 从那个 RRD 中取出指定  
  
    DS（eth0\_in、eth0\_out）的某个类型的统计值（还可以指定 resolution、start、end），并把这一切放入到一个变量 <vname>中 。  
      
    为什么还有一个 CF 字段？因为 RRA 有多种  CF 类型，有些 RRA 可能用来保存平均值、有些 RRA 可能用于统计最大值、  
      
    最小值等等。所以你必须同时指定使用什么 CF 类型的 RRA的数据。至于 :start 和 :end 、:reduce 则用得比较少，最常用的就是 :step 了，  
      
    它可以让你控制 RRDtool 从那个 RRA 中取数据。  
  
B）VDEF 是 Variable Definition （变量定义）的意思。定义什么呢？记得 MRTG 在图表的下面有一个称之为 Legend 的部分吗？  
  
    那里显示了1个或者2个 DS （MRTG 没有 DS 一说，这里是借用 RRDtool 的）的 “最大值”、“平均值”、“当前值”。      
    RRDtool 中用 VDEF 来定义。这个变量专门存放某个 DS 某种类型的值，例如 eth0\_in 的最大值、eht0\_out 的当前值等。当你需要象  
      
     MRTG  一样输出数字报表（Legend） 时，就可以在 GPRINT 子句（sub clause）中调用它。  
     
   同样它也需要用一个变量来存放数值。要注意的是，旧版 的 RRDtool 中是用另外一种格式来达到相同的目的。新版的 RRDtool 则推荐使用  
     
    VDEF   语句。但在使用过程中，却发现 VDEF 的使用反而造成了困扰。 例如你有5个 DS 要画，每个 DS 你都想输出最大值、最小值、平均值  
      
    、当前值。  如果使用 VDEF ，则需要 4 \* 5 = 20 个 VDEF 语句，这会造成极大的困扰。具体例子可以看第十一节“数字报表”部分。  
     
C）CDEF 是 Calculation Define 的意思。使用过MRTG 的都会体会到一点，MRTG 的计算能力实在太差了。例如你有两个 Target ，  
  
      一个是 eth0\_in ， 一个是 eth0\_out,如果要把它们相加起来，再除以8，得出 bytes 为单位的值，如何计算呢？或者说你只想看  
        
      eth0\_in 中超过 10Mb/s 的那部分， 其余的不关心，又如何实现呢？因为 MRTG 不能对它从 log 取出来的数据进行修改，只能原  
        
      原本本的表现，所以很难满足我们的要求。而使用 CDEF ， 这一切都可以很容易的实现。CDEF 支持很多数学运算，甚至还支持简  
        
      单的逻辑运算 if-then-else ，可以解决前面提到的第2个问题：如何只绘制你所关 心的数据。不过这一切都需要熟悉 RPN 的语法，  
        
      所以我们放到下一节介绍，这一节就介绍把 RRDtool 中的数据以图表的方式显示出来。  
**三）选项分类**本部分我们按照官方文档的方式，把选项分成几大类，分为  ：  
  
A）Time range ： 用于控制图表的X轴显示的起始/结束时间，也包括从RRA中提取指定时间的数据。  
  
B）Labels ：用于控制 X/Y 轴的说明文字。  
  
C）Size ：用于控制图片的大小。  
  
D）Limits ：用于控制 Y 轴的上下限。  
  
E）Grid ：用于控制 X/Y 轴的刻度如何显示。  
  
F）Miscellaneous ：其他选项。例如显示中文、水印效果等等。  
  
G）Report ：数字报表  
  
需要说明的是，本篇当中并不是列出了所有选项的用法，只是列出较为常用的选项，如果想查看所有选项的的用法，可以到官方站点下载文档。  
 **四）时间范围控制（Time Range）**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

[-s|--start time] [-e|--end time] [-S|--step seconds]

既然要绘图，就应该有一个起始/结束的时间。Graph 操作中有 –s ，-e 选项。这两个选项即可以用于控制图表的 X 轴显示的时间范围，也可以用于控制 RRDtool 从 RRA 中提取对应时间的数据。如果没有指定 –-end ，默认为 now；如果没有指定 –-start，则默认为1天前。如果两者都没有指定，则图表默认显示从当前算起1天内的。数  
回头看一下 DEF 中，也有 :start ,:end , :step ，这些和 –-start、--end、--step 之间有什么关系呢？  
  
让我们先看 :step 和 –step之间的关系是如何的。  
下面以 eth0.rrrd 为例，假设要绘制的时间范围 range 等于 end -start[  
A）如果 0 <  ragne < 180000 （第一个 RRA 的时间覆盖范围） ，则默认从第1个RRA中取数据 ：  
     如果 DEF 中给出的 :step > 300 ，例如 1000 ，则从 resolution= 1000 的或者第一个高于 1000 的RRA 中挑选数据，由于 eth0.rrd 中没有 resolution = 1000 的 RRA，则 RRDtool 会从 resolution = 1200 的第2 RRA 中取数据。

     如果 DEF 中给出的 :step <= 300 ，例如 200 ，则 RRDtool 会忽略该设定，还是从第一个 RRA 中取数据。  
B）如果 180000 < range < 720000  ，由于第一个RRA只能保存2天的数据，所以默认从第2个RRA中取数据 ：  
     如果 DEF 中给出的 :step > 1200 ，例如 1800，则 RRDtool 会从 resolution = 7200 的第3 RRA 中取数据  
     如果 DEF 中给出 :step<= 1200，例如 300 ，则 RRDtool 会忽略，还是从第2个 RRA 中取数据  
C）如果 720000 < range <  4320000 ，则默认从第三个 RRA 中取数据 ：  
     如果 DEF 中给出的 :step > 7200 ，例如 10000 ，则从第4个 RRA中取数据  
     如果 DEF 中给出的 :step <= 7200 ， 例如 1200 ，则忽略该值，并还是从第3 个 RRA 中取数据  
D）如果 4320000 < range < 63072000 ，则默认从第4个 RRA 中取数据 ：  
     如果 DEF 中给出的 :step > 86400 ，则行为未知  
     如果 DEF 中给出的 :step <=86400 ，则从第4个 RRA 中取数据  
E）-S 选项可以直接控制 RRDtool 挑选 RRA 。  
    例如 -S 1200 ，即使DEF 中不加 :setp ，则 RRDtool 会从第2个 RRA 中取数据，即使 rang < 180000   
    如果 -S 和 :step 同时出现，则 :step 优先。  
F）DEF 中的 :start 和 :end 可以覆盖 –-start 和 –-end 的值。  
     默认情况下，如果 DEF 中不加 :start 和 :end ，则等于 –-start 和 –end  
    如果 DEF 中定义了 :start 和 :end ，则以 :start 和 :end 为准。 **实例1 ：使用 –-start 指定 X 轴的起始时间**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

[root@dns1 bob]# rrdtool graph 1.png \          
> --start now-120000 \        # 表示起始时间是当前时间往前 120000 秒，也就是 33 个小时左右  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE \         # 从eth0.rrd 中取出 eth0\_in 的数据，CF 类型为 AVERAGE  
> AREA:value1#ff0000        # 用“方块”的形式来绘制 value1 ，注意这里是用 value1 ，不是用 eth0\_in  
481x154                        # 如果 RRDtool 有绘图方面的语句，则这里显示图片大小，否则为 0x0。  
[root@dns1 bob]#

可以看到 X 轴的文字有些是乱码，不过不要紧，你可以临时已用 env LANG=C rrdtool xxxx 来解决该问题，或者在后面用  
–n 来设定 RRDtool 使用中文字体，就不会出现这样的情况了 **实例2 ：使用 :step 从第2个RRA中取数据**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

[root@dns1 bob]# rrdtool graph 2.png \  
> --start now-120000 \  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE:step=1000 \        # :step 指定 resolution=1000   
> AREA:value1#ff0000  
481x154  
[root@dns1 bob]#

这里是 :step=1000，但 RRDtool 会取 :step=1200 的 第2个 RRA 的数据来绘图,可以和上面的 1.png 比较，发现比较平滑。

**实例3 ：使用 –S 从第2个RRA中取数据**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

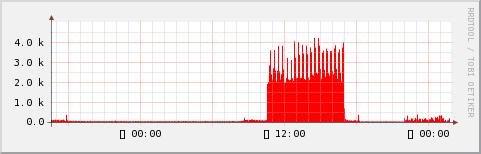
[root@dns1 bob]# rrdtool graph 4.png   
> -S 1200 \                # 使用 –S 控制 RRDtool 从 resolution=1200 的 RRA 中取数据  
> --start now-120000 \  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE \  
> AREA:value1#ff0000          
481x154  
[root@dns1 bob]#

可以看到和上面的图一样，说明 RRDtool 的确按照 -S 的设置从第2个RRA 中取数据了  
  
使用 –S 可以对 DEF 中所有的 DS 都使用相同的 resolution，等于在每个 DEF后都加上 :step=<value> ，value 是 –S 的值  
 **实例4 ：同时使用 –S 和 :step**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

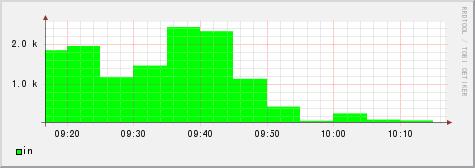
[root@dns1 bob]# rrdtool graph 5.png   
> -S 1200 \                                # -S 指定 resolution=1200  
> --start now-120000 \  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE:step=300 \        # :step 指定 resolution=300  
> AREA:value1#ff0000       
481x154  
[root@dns1 bob]#

  
  
可以看到 5.png 和 1.png 是一样的，也就是说 –S 1200 并没有起作用，而是 :step=300 起作用了  
  
**实例5 ：使用 :start 和 :end 只显示指定时间内的数据**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

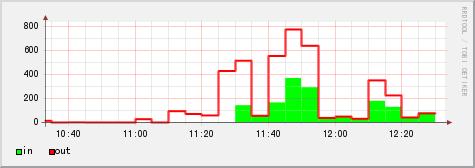
[root@dns1 bob]# rrdtool graph 1.png \  
> --start now-1h \                        # X 轴显示1个小时的长度  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE:start=now-600:end=now-300 \        # 但只取10分钟前到5分钟前的这部分  
> AREA:value1#00ff00:in    
475x168  
[root@dns1 bob]#

  
如果我们不加 :start 和 :end ，则效果如下 ：  
  
  
我们甚至可以让两个对象显示不同的时间，例如   
  
**实例6 ：让两个对象显示不同时间段的数据**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

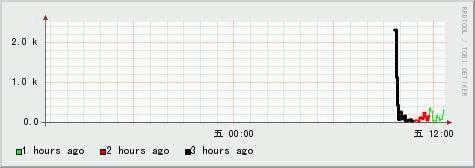
[root@dns1 bob]# rrdtool graph 1.png \  
> --start now-2h \                # 规定时间为2小时内  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE:end=now:start=end-1h \        # 规定时间为1小时内                  
> DEF:value2=eth0.rrd:eth0\_out:AVERAGE \        # 没有指定 :start 和 :end，默认和 –-start 一样也是2小时  
> AREA:value1#00ff00:in \  
> LINE2:value2#ff0000:out:STACK   
475x168  
[root@dns1 bob]#

  
 **实例7 ：把一段时间分为几段分别显示 ：**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864875)

[root@dns1 bob]# rrdtool graph 1.png \  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE:end=now:start=end-1h \        # 当前1小时内  
> DEF:value2=eth0.rrd:eth0\_in:AVERAGE:end=now-1h:start=now-2h \        # 2小时前  
> DEF:value3=eth0.rrd:eth0\_in:AVERAGE:end=now-2h:start=now-3h \        # 3小时前  
> LINE1:value1#00ff00:"1 hours ago" \  
> LINE2:value2#ff0000:"2 hours ago" \  
> LINE3:value3#000000:"3 hours ago"    
475x168  
[root@dns1 bob]

  
  
我 们把3个小时内的数据用三种不同粗细、不同颜色的曲线画了出来。out部分（红色）显示了2个小时内的流量，而in部分（绿色）则只显示了1个小时内的部 分。在这里要提一点，虽然我们指定了 –-start 或者 –-end ,或者 :start , :end，但并不意味着曲线就一定会从指定的时间点开始和结束。  
例如我们上面指定了 :start=now-600:end=now-300 ，也就是只显示5分钟的数据。但图表出来的效果却是10(10:05-10:15)分钟的数据，这是因为我们挑选的时间当中“不慎”横垮了两个周期 (10:05-10:10,10:10-10:15)，所以 RRDtool 会把它们全部画出来，而不是只画其中的5分钟。   
 **五）说明文字（Label）**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

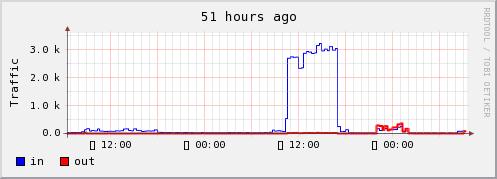
[-t|--title string] [-v|--vertical-label string]

-t 是用于图表上方的标题，-v 是用于 Y 轴的说明文字  
  
**实例1 ：给图表增加标题**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

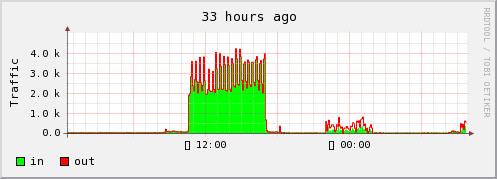
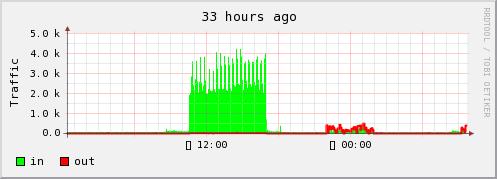
[root@dns1 bob]# rrdtool graph 1.png \  
> --start now-183600 \                # 从当前开始往前51个小时  
> -t "51 hours ago" -v "Traffic" \        # 标题是 “51 hours ago”，Y 轴的说明文字是 “Traffic”  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE \  
> DEF:value2=eth0.rrd:eth0\_out:AVERAGE \  
> LINE1:value1#0000ff:in \        # 注释 ：以1个像素宽的曲线画出 value1，颜色是蓝色，图例的说明文字是“in”  
> LINE2:value2#ff0000:out        # 注释 ：以2个像素宽的曲线画出 value2，颜色是红色，图例的说明文字是 “out”  
497x179  
[root@dns1 bob]#

  
  
现在我们用的是 LINE 的方式来绘图。LINE 可以有3种，分别是 LINE1|2|3,也就是线条的粗细。还有一种是 STACK 方式下面再介绍。  
可以看到流入的流量比流出的流量稍大，这样看的话，out 流量比较难看，是否可以有别的方式呢？RRDtool 还提供了另外一种格式，就是 STACK 。意思就是在前一个对象的基础（把前一个对象的值当成 X 轴）上绘图，而不是从 X 轴开始。  
  
**实例2 ：使用 STACK 方式绘图**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

[root@dns1 bob]# rrdtool graph 3.png \  
> --start now-120000 \  
> -t "33 hours ago" \  
> -v "Traffic" \  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE \  
> DEF:value2=eth0.rrd:eth0\_out:AVERAGE \  
> AREA:value1#00ff00:in \  
> LINE:value2#ff0000:out:STACK                # 注意最后的 “STACK” ，表示在 value1 的基础上绘图  
497x179  
[root@dns1 bob]#

  
这是没有采用 STACK 方式绘图的效果 ：  
  
  
可 以看得出上面采用 STACK 方式的比较清晰，但要注意，采用 STACK 方式后，在读取 out 流量时，Y 轴的刻度不再是 out 的值，应该用刻度值减去 in 的值，才是 out 真正的值。这点比较麻烦。需要配合 GPRINT 语句才能达到一定的效果。  
  
**六）图表大小（Size）**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

[-w|--width pixels] [-h|--height pixels]

这里说图表大小而不是图片大小，是因为 –w ，-h 控制的是 X/Y 轴共同围起来的那部分大小，而不是整个图片的大小，这点在前面就可以看出了。  
默认的图表大小是 400 （长）x 100 （高），但一般会返回497x179 这样的数字，这个才是图片的大小。  
RRDtool 比 MRTG 好的一个地方就是 MRTG 一放大图片，就会变得模糊。RRDtool 则不会。  
在官方文档中，-w 似乎是一个比较敏感的参数，我们看下面的描述 ：

**QUOTE:**

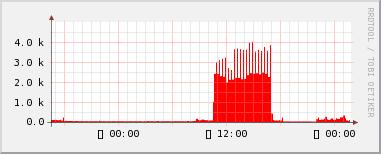
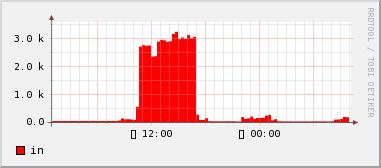
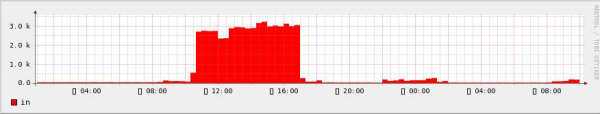
First it makes sure that the RRA covers as much of the graphing time frame as possible. Second it looks at the resolution of the  
  
RRA compared to the resolution of the graph. It tries to find one which has the same or higher better resolution. With the ``-r‘‘   
  
option you can force RRDtool to assume a different resolution than the one calculated from the pixel width of the graph.

**实例1 ：使用 –w 设定图表大小为 300 像素**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

[root@dns1 bob]# rrdtool graph 3.png   
> -w 300 \                                # 设定 size 为 300 pixel  
> --start now-120000 \  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE \  
> AREA:value1#ff0000             
381x154  
[root@dns1 bob]#

  
  
可 以看到图表是不是变小了呢？而且整个图片的大小也变小了。如果用前面的话来推理，由于 120000/300（-w的值）= 400 > 300 （step）,由于没有 resolution=400 的 RRA，所以应该采用 resolution=7200 的第2个 RRA 的数据来绘图，但实际上不是。  
  
  
  
上面这个才是 300 pixel 宽，resolution=7200 的效果所以–w 和 –h 并不能影响 RRDtool 选择 RRA ，只能起到缩小放大的作用。BTW：当你绘制的时间范围较大时，可以使用 –w 增大图表大小  
  
  
 **七） Y 轴上下限（Limits）**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

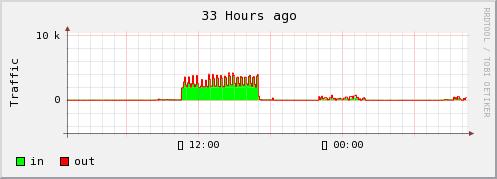
[-u|--upper-limit value] [-l|--lower-limit value] [-r|--rigid]

默认情况下，RRDtool 会和 MRTG 一样自动调整 Y 轴的数字，来配合当前的数值大小。如果想禁止该特性，可以通过 –upper-limit  
-–lower-limit 来做限制，表示Y轴显示的值从多少到多少。如果没有指定 –rigid ，则在图表的上下边界处还是会有一些延伸，但如果指定了   
-–rigid ，则严格按照 –-upper-limit 和 –-lower-limit 绘制。  
在使用 –lower-limit 时要注意，如果数据中有负数，如果 -–lower-limit 为 0，则负数部分是显示不出来的。  
  
**实例1 ：使用 –-upper-limit 和 –-lower-limit 限制 Y 轴的上下限**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

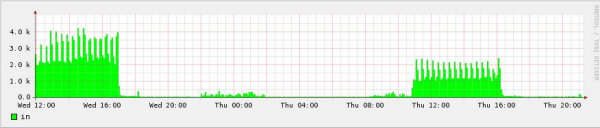
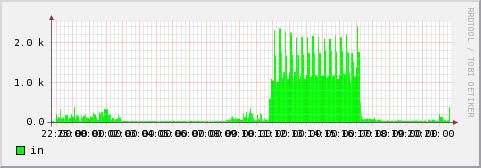
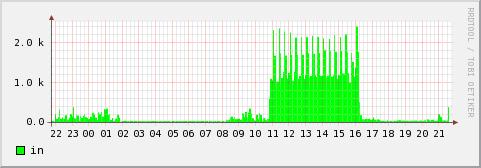
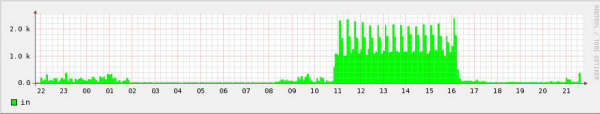
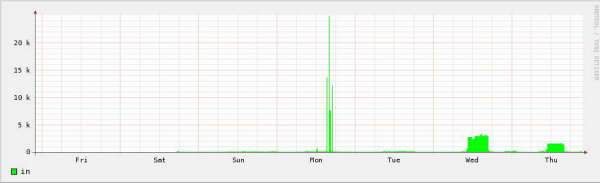
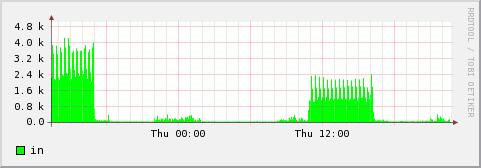
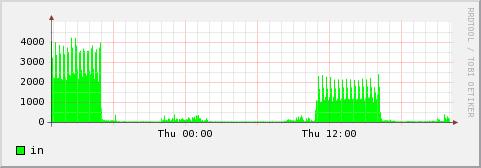
[root@dns1 bob]# rrdtool graph 1.png \  
> --start now-120000 \  
> -v "Traffic" -t "33 Hours ago" \  
> --lower-limit -5000 \                # 限制Y轴下限为 -5000   
> --upper-limit 10000 \                # 上限为 10000  
> --rigid \                                         # 严格按照上面的规定来画  
> DEF:value1=eth0.rrd:eth0\_in:AVERAGE \  
> DEF:value2=eth0.rrd:eth0\_out:AVERAGE \  
> AREA:value1#00ff00:in \  
> LINE1:value2#ff0000:out:STACK  
497x179  
[root@dns1 bob]#

  
 **八） X/Y 轴刻度（Grid）**

**CODE:**

[[Copy to clipboard]](http://bbs.chinaunix.net/viewthread.php?tid=864876)

[-x|--x-grid GTM:GST:MTM:MST:LTM:LST:LPR:LFM]   
[-x|--x-grid none]  
[-y|--y-grid grid step:label factor]   
[-y|--y-grid none]  
[-Y|--alt-y-grid]  
[-X|--units-exponent value]

RRDtool 中设置 X 轴的刻度比较复杂，如果没有必要，可以交给 RRDtool 自动去处理。  
例如上面的图，33 小时的情况下，X 轴只有2个值，显得很不足。这时候有两种方法 ：  
A）一是使用 –w 增大图表的宽度，这样 RRDtool 会自动加多一些刻度上去。  
  
  
  
B） 二是通过上面的选项自己设置 X/Y 轴的刻度如何显示。首先看上图，在垂直的线中，红色的线称为 Major Grid（主要网格线），灰色的线称为 Base Grid （次要网格线）（这里是借用 EXCEL 中的概念）。 X 轴下面的时间文字成为 Label 。      
  
C）有两种方法可以快速去掉 X/Y 轴的刻度，就是 –-x-grid none 和 –-y-grid none  
  
D）GTM:GST ：控制次要格网线的位置。GTM 是一个时间单位，可以是 SECOND、MINUTE、HOUR、DAY 、WEEK、MONTH、YEAR 。  
  
     GST 则是一个数字，控制每隔多长时间放置一根次要格线。例如我们要画一个1天的图表，决定每15分钟一根次要网格线，则格式为 MINUTE:15  
       
E）MTM:MST ：控制主要网格线的位置。MTM 同样是时间单位，MST 是一个数字。接上面的例子，决定一个小时1根主要网格线。则格式为 HOUR:1   
  
     LTM:LST ：控制每隔多长时间输出一个label 。决定为1小时1个 label 。则格式为 HOUR:1  
  
G）LPR:LFM ：LTM:LST 只是决定了 label 的显示位置了，没有指定要显示什么内容。LPR 指的是如何放置 label 。如果LPR 为0，则数字对齐格线  
  
    （适用于显示时间）。如果不为0，则会做一些偏移（适用于显示星期几、月份等）。至于LFM 则需要熟悉一下 date 命令的参数，常用的有 %a（星期几）、  
      
    %b（月份）、%d（天）、%H（小时）、%M（分）、%Y（年）。我们决定显示小时和分，所以用 %H%M  
  
H）综合起来，X 轴的刻度定义就是 –-x-grid MINUTE:15:HOUR:1:HOUR:1:0:’%H:%M’。最好把 %H:%M 括起来  
  
     建议 MST是 GST 的2-6倍，MST 和 LST 相同。这样画出来的图比较美观一些  
  
这明显就是图片太小了，不够显示。把上面的 :%M 去掉就可以了，只显示小时，不显示分钟  
  
如果把图片放大一点就更好了 (-w 800)  
  
所以在设置 X 轴的刻度时，要记得不要显示太多东西，否则需要增大图片的大小  
  
  
I）Y 轴刻度的设置又不一样了  
  
   grid step ：用于控制Y轴每隔多少显示一根水平线  
  
   label factor ：默认为1，也就是在每根水平线的高度那里显示一个值。  
  
   例如下面就是一个例子 （每隔800显示一根水平线）  
  
  
J）Y 轴还有一个很方便的选项就是 –Y ，它可以最大限度的优化 Y 轴的刻度，建议每次绘图都加上去。  
  
K）Y 轴另外一个有用的选项就是 –X （虽然选项名是 -X ，但确实是用来设置 Y 轴刻度值的）。在上面的图我们看到 RRDtool 自动对 Y 轴的值进行调整，  
  
     以 k 为单位显示。但如果你不想以 k 显示，而是想固定以某个单位来显示（M，b）该怎么办呢？这就要用到 –X 选项了。-X 后面跟一个参数，参数值  
       
     范围是 -18、-15、-12、-9、-6 、-3、0、3、6、9、12、15、18 。0 表示以原值显示，3 表示数值除以1000，也就是以 k 为单位显示，6 就是以  
       
     M 显示，9 就是以 G 显示。如果你给出1或者2，则 RRDtool 也可以接受，但会被“静悄悄”的改为0。下面就是一个以原值（-X 0）显示的例子  


### RRDTool 详解 - Share your knowledge … - 51CTO技术博客

http://freeloda.blog.51cto.com/2033581/1307492

原创作品，允许转载，转载时请务必以超链接形式标明文章 [原始出处](http://freeloda.blog.51cto.com/2033581/1307492) 、作者信息和本声明。否则将追究法律责任。<http://freeloda.blog.51cto.com/2033581/1307492>

大纲

一、MRTG的不足和RRDTool对比

二、RRDTool概述

三、安装RRDTool

四、RRDTool绘图步骤

五、rrdtool命令详解

六、RRDTool绘图案例

注，实验环境 CentOS 6.4 x86\_64，软件版本 rrdtool-1.3.8(最新版的是1.4.8)。(说明：本博文的一些图片和内容自于开源社区与官方网站并不是所有内容全是原创)

一、MRTG的不足和RRDTool对比

1.MRTG软件的不足

MRTG监测网络流量很方便，但是如果需要监测其他的系统性能，比如CPU负载、系统负载、网络连接数等，就不是那么简单了。即使实现了这些功能，但管理起来非常麻烦。比如公司有一两千个被监测点，分布在不同的机房，为了管理方便需要将这些服务器和网络设备分类，这样的话就需要将这些被监测点放在不同的MRTG配置文件中，运行多个crontab，甚至还要自己写HTML页面对其进行管理。

MRTG毕竟是一套很老的软件，而且存在许多不足的地方，其作者Tobias Oetiker在1999年就已经开始开发另一套开源软件RRDTool来代替MRTG。现在RRDTool已经发展得很成熟，在功能MRTG难以与其相提并论。

2.RRDTool与MRTG对比

与MRTG一样，RRDTool也是由Tobias Oetiker撰写的开源软件，但RRDTool并非MRTG的升级版本，两者有非常大的区别，也可以说RRDTool是将用来取代MRTG的产品。下面是两个软件的一些优缺点的对比。

(1).MRTG

**优点**：

* 简单、易上手，基本安装完了之后只要修改一下配置文件即可使用。

**缺点**：

* 使用文本式的数据库，数据不能重复使用。
* 只能按日、周、月、年来查看数据。
* 由于MRTG本来只是用来监测网络的流量，所以只能存储两个DS（Data Source），即存储流量的输入和输出。
* 每取一次数据即需要绘图一次，浪费系统资源。
* 图像比较模糊。
* 无用户、图像管理功能。
* 没有详细日志系统。
* 无法详细了解各流量的构成。
* 只能用于TCP/IP网络，对于SAN网络流量无能为力。
* 不能在命令行下工作。

(2).RRDTool

**优点：**

* 使用RRD（Round Robin Database）存储格式，数据等于放在数据库中，可以方便地调用。比如，将一个RRD文件中的数据与另一个RRD文件中的数据相加。
* 可以定义任意时间段画图，可以用半年数据画一张图，也可以用半小时内的数据画一张图。
* 能画任意个DS，多种图形显示方式。
* 数据存储与绘图分开，减轻系统负载。
* 能任意处理RRD文件中的数据，比如，在浏览监测中我们需要将数据由Bytes转化为bits，可以将原始数据乘8。

**缺点：**

* RRDTool的作用只是存储数据和画图，它没有MRTG中集成的数据采集功能。
* 在命令行下的使用非常复杂，参数极多。
* 无用户、图像管理功能。

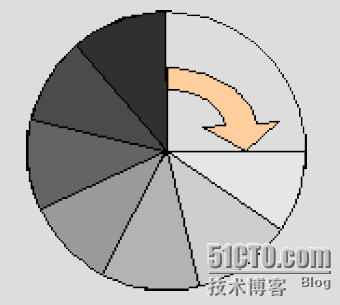
**RRDTool官方网站：**

<http://oss.oetiker.ch/rrdtool/>

二、RRDTool概述

1.概述

RRDtool 代表 “Round Robin Database tool” ，作者同时也是 MRTG 软件的发明人。官方站点位于<http://oss.oetiker.ch/rrdtool/> 。 所谓的“Round Robin” 其实是一种存储数据的方式，使用固定大小的空间来存储数据，并有一个指针指向最新的数据的位置。我们可以把用于存储数据的数据库的空间看成一个圆，上面有很多刻度。这些刻度所在的位置就代表用于存储数据的地方。所谓指针，可以认为是从圆心指向这些刻度的一条直线。指针会随着数据的读写自动移动。要注意的是，这个圆没有起点和终点，所以指针可以一直移动，而不用担心到达终点后就无法前进的问题。在一段时间后，当所有的空间都存满了数据，就又从头开始存放。这样整个存储空间的大小就是一个固定的数值。所以RRDtool 就是使用类似的方式来存放数据的工具， RRDtool 所使用的数据库文件的后缀名是'.rrd。如下图，

[](http://img1.51cto.com/attachment/201310/11/2033581_1381480511i8J1.png)

2.特点

* 首先 RRDtool 存储数据，扮演了一个后台工具的角色。但同时 RRDtool 又允许创建图表，这使得RRDtool看起来又像是前端工具。其他的数据库只能存储数据，不能创建图表。
* RRDtool 的每个 rrd 文件的大小是固定的，而普通的数据库文件的大小是随着时间而增加的。
* 其他数据库只是被动的接受数据， RRDtool 可以对收到的数据进行计算，例如前后两个数据的变化程度（rate of change），并存储该结果。
* RRDtool 要求定时获取数据，其他数据库则没有该要求。如果在一个时间间隔内（heartbeat）没有收到值，则会用 UNKN (unknow)代替，其他数据库则不会这样。

三、安装RRDTool

1.安装yum源

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# rpm -ivh http://download.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm  [root@node1 ~]# rpm -ivh http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.3-1.el6.rf.x86\_64.rpm |

2.同步时间

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# ntpdate 202.120.2.101 |

3.下载rrdtool的RPM包

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# wget ftp://195.220.108.108/linux/centos/6.4/os/x86\_64/Packages/rrdtool-1.3.8-6.el6.x86\_64.rpm |

4.yum安装rrdtool

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# yum -y localinstall --nogpgcheck rrdtool-1.3.8-6.el6.x86\_64.rpm  yum install –y rrdtool |

5.查看一下安装文件

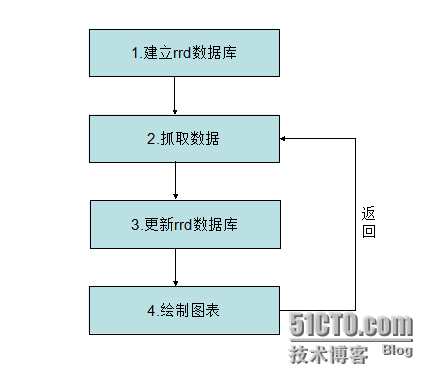
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | [root@node1 ~]# rpm -qa | grep rrdtool  rrdtool-1.3.8-6.el6.x86\_64  [root@node1 ~]# rpm -ql rrdtool  /usr/bin/rrdcgi  /usr/bin/rrdtool #命令行工具  /usr/bin/rrdupdate  /usr/lib64/librrd.so.4 #下面是库文件  /usr/lib64/librrd.so.4.0.7  /usr/lib64/librrd\_th.so.4  /usr/lib64/librrd\_th.so.4.0.7  /usr/share/man/man1/bin\_dec\_hex.1.gz #下面是帮助文档  /usr/share/man/man1/cdeftutorial.1.gz  /usr/share/man/man1/rpntutorial.1.gz  /usr/share/man/man1/rrd-beginners.1.gz  /usr/share/man/man1/rrdbuild.1.gz  /usr/share/man/man1/rrdcgi.1.gz  /usr/share/man/man1/rrdcreate.1.gz  /usr/share/man/man1/rrddump.1.gz  /usr/share/man/man1/rrdfetch.1.gz  /usr/share/man/man1/rrdfirst.1.gz  /usr/share/man/man1/rrdgraph.1.gz  /usr/share/man/man1/rrdgraph\_data.1.gz  /usr/share/man/man1/rrdgraph\_examples.1.gz  /usr/share/man/man1/rrdgraph\_graph.1.gz  /usr/share/man/man1/rrdgraph\_rpn.1.gz  /usr/share/man/man1/rrdinfo.1.gz  /usr/share/man/man1/rrdlast.1.gz  /usr/share/man/man1/rrdlastupdate.1.gz  /usr/share/man/man1/rrdresize.1.gz  /usr/share/man/man1/rrdrestore.1.gz  /usr/share/man/man1/rrdthreads.1.gz  /usr/share/man/man1/rrdtool.1.gz  /usr/share/man/man1/rrdtune.1.gz  /usr/share/man/man1/rrdtutorial.1.gz  /usr/share/man/man1/rrdupdate.1.gz  /usr/share/man/man1/rrdxport.1.gz  /usr/share/rrdtool |

6.查看一下命令行工具

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [root@node1 ~]# rrdtool -h  RRDtool 1.3.8 Copyright 1997-2009 by Tobias Oetiker <tobi@oetiker.ch>          Compiled Aug 21 2010 10:57:18  Usage: rrdtool [options] command command\_options  Valid commands: create, update, updatev, graph, graphv, dump, restore,      last, lastupdate, first, info, fetch, tune,      resize, xport  RRDtool is distributed under the Terms of the GNU General  Public License Version 2. (www.gnu.org/copyleft/gpl.html)  For more information read the RRD manpages |

注，使用 man rrdtool 可以查看详细的使用方法。好了，到这里我们就安装完成了，下面我们来说一下RRDTool的绘图步骤。

四、RRDTool绘图步骤

[](http://img1.51cto.com/attachment/201310/11/2033581_1381480511eweV.png)

步骤一︰建立RRD文件，这个文件说来就是RRDtool的专属数据库。RRDtool以自有的格式存放数据。下面会讲解！

步骤二︰抓取数据个人觉得是整个RRDtool最困难的一部分，因为RRDtool的数据是要靠自己在创建RRD数据库时定义出来，不像MRTG内建抓数据功能，但是却因为如此，可以给RRDtool画图的数据弹性也比较大，例如︰snmp查询结果、系统状态、网页中特定数据统计等等。

步骤三︰将抓下来的数据就用rrdtool update的指令进行更新到的RRD数据库中，让图表能画出最新的流量。

步骤四︰这就是重点啦！通过rrdtool graph的指令来依据RRD数据库的数据进行绘图，这也是使用者唯一看的到的东西，若规划的不好会影响使用者阅读上的困难！

循环︰由于要完成动态绘图的图表，第二步骤到第四步骤必须不断的重复执行以维持资料的更新，目前知道要达成循环的方法有两种︰

* 在Script中使用循环
* 使用crontab任务计划

五、rrdtool命令详解

1.创建RRD数据库

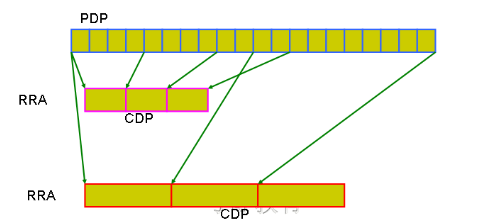
create 语法

|  |  |
| --- | --- |
| 1  2  3  4  5 | rrdtool create filename  [--start|-b start time]  [--step|-s step]  DS:ds-name:DST:dst arguments #最后获取的数据是PDP，更新数据时要考滤DS顺序(\*把所有要更新的数据，按照DS定义的顺序用冒号格开\*)  RRA:CF:cf arguments #最后获取的数据是CDP,绘图时使用的是这些数据 |

参数详解：

* DS：DS 用于定义 Data Soure 。也就是用于存放结果的变量名。DS是用来申明数据源的，也可以理解为申明数据变量，也就是你要检测的端口对应的变量名，这个参数在画图的时候还要使用的。
* DST：DST 就是DS的类型。有 COUNTER、GUAGE、DERIVE、ABSOLUTE、COMPUTE 5种。由于网卡流量属于计数器型，所以这里应该为 COUNTER 。
* RRA：RRA 用于指定数据如何存放。我们可以把一个RRA 看成一个表，各保存不同 interval 的统计结果。RRA的作用就是定义更新的数据是如何记录的。比如我们每5分钟产生一条刷新的数据，那么一个小时就是12条。每天就是288条。这么庞大的数据量，一定不可能都存下来。肯定有一个合并（consolidate）数据的方式，那么这个就是RRA的作用了。
* PDP：Primary Data Point 。正常情况下每个 interval RRDtool 都会收到一个值；RRDtool 在收到脚本给来的值后会计算出另外一个值（例如平均值），这个 值就是 PDP ；这个值代表的一般是“xxx/秒”的含义。注意，该值不一定等于RRDtool 收到的那个值。除非是GAUGE ，可以看下面的例子就知道了
* CF：CF 就是 Consolidation Function 的缩写。也就是合并（统计）功能。有 AVERAGE、MAX、MIN、LAST 四种分别表示对多个PDP 进行取平均、取最大值、取最小值、取当前值四种类型。具体作用等到 update 操作时再说。
* CDP：Consolidation Data Point 。RRDtool 使用多个 PDP 合并为（计算出）一个 CDP。也就是执行上面 的CF 操作后的结果。这个值就是存入 RRA的数据，绘图时使用的也是这些数据

下面是RRA与PDP、CDP之间的关系图，

[](http://img1.51cto.com/attachment/201310/164103483.png)

(0).filename

默认是以 .rrd 结尾，但也以随你设定。

(1).--start|-b start time

设定RRD数据库加入的第一个数据值的时间，从1970-01-01 00:00:00 UTC时间以来的时间(秒)。RRDtool不会接受早于或在指定时刻上的任何数值。默认值是now-10s；如果 update 操作中给出的时间在 –-start 之前，则 RRDtool拒绝接受。--satrt 选项也是可选的。 如果你想指定--start 为1天前，可以用CODE:--start $(date -d '1 days ago' +%s)。注意，--start 选项的值必是 timestamp 的格式。

(2).--step|-s step

指定数据将要被填入RRD数据库的基本的时间间隔。默认值是300秒；

(3).DS:ds-name:DST:dst arguments DS(Data Source)

DS：DS 用于定义 Data Soure 。也就是用于存放结果的变量名。 DS是用来申明数据源的，也可以理解为申明数据变量，也就是你要检测的端口对应的变量名，这个参数在画图的时候还要使用的。这里开始定义RRD数据的基本属性；单个RRD数据库可以接受来自几个数据源的输入。在DS选项中要为每个需要在RRD中存储的数据源指定一些基本的属性；ds-name数据域命名；DST定义数据源的类型，dst arguments参数依赖于数据源的类型。

案例：DS:mysql:COUNTER:600:0:100000000

DS(Data Source，数据源)表达式总共有六个栏位：

* DS 表示这个为DS表达式
* ds-name 数据域命名
* DST 定义数据源的类型
* heartbeat 有效期(heartbeat)，案例里的值为'600'，假设要取12:00的数据，而前后300秒里的值(11:55-12:05)经过平均或是取最大或最小都算是12:00的有效值；
* min 允许存放的最小值，此例允许最小为0。
* max 允许存放的最大值，最大为100000000。

注，如果不想设限制可以再第五个栏位和第六个栏位以 "U:U"表示（U即Unknown）。

DST定义数据源的类型。数据源项的后续参数依赖于数据源的类型。对于GAUGE、COUNTER、DERIVE、以及ABSOLUTE，其数据源的格式为： DS:ds-name:GAUGE | COUNTER | DERIVE | ABSOLUTE:heartbeat:min:max。DST 的选择是十分重要的，如果选错了 DST ，即使你的脚本取的数据是对的，放入 RRDtool 后也是错误的，更不用提画出来的图是否有意义了。

* GAUGE ：GAGUE 和上面三种不同，它没有“平均”的概念，RRDtool 收到值之后字节存入 RRA 中。
* COUNTER ：必须是递增的，除非是计数器溢出。在这种情况下，RRDtool 会自动修改收到的值。例如网络接口流量、收到的packets 数量都属于这一类型。
* DERIVE：和 COUNTER 类似。但可以是递增，也可以递减，或者一会增加一会儿减少。
* ABSOLUTE ：ABSOLUTE 比较特殊，它每次都假定前一个interval的值是0，再计算平均值。
* COMPUTE ：COMPUTE 比较特殊，它并不接受输入，它的定义是一个表达式，能够引用其他DS并自动计算出某个值。例如CODE：DS:eth0\_bytes:COUNTER:600:0:U DS:eth0\_bits:COMPUTE:eth0\_bytes,8,\* 则 eth0\_bytes 每得到一个值，eth0\_bits 会自动计算出它的值：将 eth0\_bytes 的值乘以 8 。不过 COMPUTE 型的 DS 有个限制，只能应用它所在的 RRD 的 DS ，不能引用其他 RRD 的 DS。 COMPUTE 型 DS 是新版本的 RRDtool 才有的，你也可以用 CDEF 来实现该功能。如:CDEF:eth0\_bits=eth0\_bytes,8,\*

DST 实例说明，

Values = 300, 600, 900, 1200

#假设 RRDtool 收到4个值，分别是300，600，900，1200。   
Step = 300 seconds

#step 为 300   
COUNTER = 1，1，1，1

#（300-0）/300，（600-300）/300，（900-600）/300，（1200-900）/300 ，所以结果为 1，1，1，1   
DERIVE = 1，1，1，1 # 同上   
ABSOLUTE = 1，2，3，4

#(300-0)/300，(600-0)/300，(900-0)/300，(1200-0)/300，所以结果为 1，2，3，4   
GAUGE = 300，600，900，1200 # 300 , 600 ,900 ,1200 不做运算，直接存入数据库。所以第一行的 values 并不是 PDP，后面4行才是PDP。

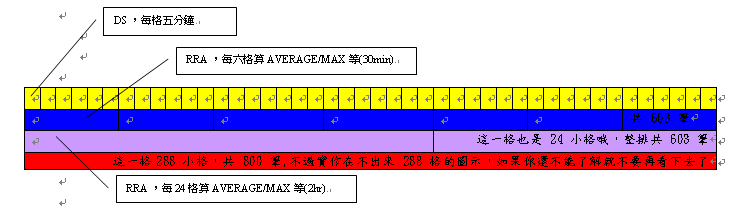
下面我们来建立一个RRD库文件，

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [root@node1 ~]# rrdtool create eth0.rrd \  > --step 300 \  > DS:eth0\_in:COUNTER:600:0:12500000 \   # 600 是 heartbeat；0 是最小值；12500000 表示最大值；  > DS:eth0\_out:COUNER:600:0:12500000 \  # 如果没有最小值/最大值，可以用 U 代替，例如 U:U  > RRA:AVERAGE:0.5:1:600 \  # 1 表示对1个 PDP 取平均。实际上就等于 PDP 的值  > RRA:AVERAGE:0.5:4:600 \  # 4 表示每4个 PDP 合成为一个 CDP，也就是20分钟。方法是对4个PDP取平均，  > RRA:AVERAGE:0.5:24:600 \ # 同上，但改为24个，也就是24\*5=120分钟=2小时。  > RRA:AVERAGE:0.5:288:730   # 同上，但改为288个，也就是 288\*5=1440分钟=1天  [root@node1 ~]# ll -h eth0.rrd  -rw-r--r--  1 root   root     41K 10月 11 10:16 eth0.rrd |

有的人可能会问，上面有两个 DS，那 RRA 中究竟存的是那个 DS 的数据呢？实际上，这些 RRA 是共用的，你只需建立一个 RRA，它就可以用于全部的 DS 。所以在定义 RRA 时不需要指定是给那个 DS 用的。

(4).RRA:CF:cf arguments

RRA的作用就是定义更新的数据是如何记录的。比如我们每5分钟产生一条刷新的数据，那么一个小时就是12条。每天就是288条。这么庞大的数据量，一定不可能都存下来。肯定有一个合并（consolidate）数据的方式，那么这个就是RRA的作用了。如下图，

[](http://img1.51cto.com/attachment/201310/11/2033581_1381480512CWPT.png)

RRD的一个目的是在一个环型数据归档中存储数据。一个归档有大量的数据值或者每个已定义的数据源的统计，而且它是在一个RRA行中被定义的。当一个数据进入RRD数据库时，首先填入到用 -s 选项所定义的步长的时隙中的数据，就成为一个pdp值，称为首要数据点（Primary Data Point）。该数据也会被用该归档的CF归并函数进行处理。可以把各个PDPs通过某个聚合函数进行归并的归并函数有这样几种：AVERAGE、MIN、MAX、LAST等。这些归并函数的RRA命令行格式为:RRA:AVERAGE | MIN | MAX | LAST:xff:steps:rows。

什么是 CF？

以上面的案例中第2个RRA 和 4，2，1，3 这4个 PDP 为例

* AVERAGE ：则结果为 (4+2+1+3)/4=2.5
* MAX ：结果为4个数中的最大值 4
* MIN ：结果为4个数中的最小值1
* LAST ：结果为4个数中的最后一个 3

同理，第三个RRA和第4个RRA则是每24个 PDP、每288个 PDP 合成为1个 CDP。

解释度（Resolution）

这里要提到一个 Resolution 的概念，在官方文档中多处提到 resolution 一词。Resolution 究竟是什么？Resolutino 有什么用？举个例子，如果我们要绘制1小时的数据，也就是60分钟，那么我们可以从第一个RRA 中取出12个 CDP 来绘图；也可以从第2个 RRA中取出3个 CDP 来绘图。到底 RRDtool 会使用那个呢？让我们看一下 RRA 的定义 ：RRA:AVERAGE:0.5:4:600 。

Resolution 就等于 4 \* step = 4 \* 300 = 1200 ，也就是说 ，resolution 是每个CDP 所代表的时间范围，或者说 RRA 中每个 CDP（记录）之间的时间间隔。所以第一个 RRA 的 resolution 是 1\* step=300，第2是 1200，第三个是 24\*300=7200，第4个 RRA 是 86400 。

默认情况下，RRDtool 会自动挑选合适的 resolution 的那个 RRA 的数据来绘图。我们大可不必关心它。但如果自己想取特定 RRA 的数据，就需要用到它了。关于 Resolution 我们还会在 fetch 和 graph 中提到它。

xff 字段

细心的朋友可能会发现，在 RRA 的定义中有一个数值，固定是 0.5 ，这个到底是什么东东呢？   
这个称为 xff 字段，是 xfile factor 的缩写。让我们来看它的定义 ：

QUOTE:

The xfiles factor defines what part of a consolidation interval may be made up from \*UNKNOWN\* data while   
the consolidated value is still regarded as known. It is given as the ratio of allowed \*UNKNOWN\* PDPs to   
the number of PDPs in the interval. Thus, it ranges from 0 to 1 (exclusive)

这个看起来有点头晕，我们举个简单的例子 ：例如

CODE:RRA:AVERAGE:0.5:24:600

这个 RRA 中，每24个 PDP （共两小时）就合成为一个 CDP，如果这 24 个 PDP 中有部分值是 UNKNOWN （原因可以很多），例如1个，那么这个 CDP合成的结果是否就为 UNKNOWN 呢？

不是的，这要看 xff 字段而定。Xff 字段实际就是一个比例值。0.5 表示一个 CDP 中的所有 PDP 如果超过一半的值为 UNKNOWN ，则该 CDP 的值就被标为UNKNOWN。也就是说，如果24个 PDP中有12个或者超过12个 PDP 的值是 UNKNOWN ，则该 CPD 就无法合成，或者合成的结果为 UNKNOWN；如果是11个 PDP 的值为 UNKNOWN ，则该 CDP 的值等于剩下 13 个 PDP 的平均值。

如果一个 CDP 是有2个 PDP 组成，xff 为 0.5 ，那么只要有一个 PDP 为 UNKNOWN ，则该 PDP 所对应的 CDP 的值就是 UNKNOWN 了。

2.抓取数据

简单说，就是用shell写个脚本去不断的收集数据。对于不懂shell编辑的博友可以去网上找点资料学习一下，很简单的。下面是利用snmp来获取进入网卡的流量。

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# snmpget -v 2c -c public 192.168.18.201 ifInOctets.2  IF-MIB::ifInOctets.2 = Counter32: 57266195 |

上面的例子是使用snmpget来抓取192.168.18.201的网卡输入流量，-v 2c表示snmp版本号，-c public snmp共同体名称，192.168.18.201是这台主机的IP地址，ifInOctets.2是指eth0网卡的输入流量，ifInOctets.1是指lo0网卡的输入流量。从上面我们可以看出，eth0输入的流量为57266195。下面我们来截取一下输入流量，

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# snmpget -v 2c -c public 192.168.18.201 ifInOctets.2 | sed -e 's/.\*ter32: \(.\*\)/\1/'  57463513 |

我们得到的值为57463513，这就是我们要的结果。我们只要用shell脚本写个循环就可以收集网卡的输入流量了，再用rrdtool update命令将收集到的数据更新到RRD数据库中即可。

当然，您不一定要使用snmpget，也可以使用snmpwalk、tcpdump等等抓取数据回来分析，说夸张点，凡是有变化的数据都可以经过处理变成我们要的资料，然后画成图表。

3.更新RRD数据库数据

update 语法

|  |  |
| --- | --- |
| 1  2  3  4 | rrdtool update filename [--template|-t ds-name[:dsname]...] N|timestamp:value[:value...]  filename RRD数据库文件名称  --template|-t ds-name[:ds-name] 要更新RRD数据库中数据源的名称，其中-t指定数据源的顺序  N|timestamp:value[:value...] 时间:要更新的值 |

案例：

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]#rrdtool update eth0.rrd 1381467942:60723022 或  [root@node1 ~]# rrdtool update eth0.rrd N:60723022 |

其中，1381467942是当前的时间戳，可以用date +%s命令获得，或者直接用N代替。60723022是当前要更新的流量数据，可以用shell脚本获得。下面我们来查看一下，更新的数据。

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# rrdtool fetch eth0.rrd AVERAGE |

4.绘制图表

使用RRDtool 我们最关心什么？当然是把数据画出来了。虽然前面谈了很多，但这些都是基础来的。掌握好了，可以让你在绘图时更加得心应手。

graph 语法

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | rrdtool graph filename [option ...]  [data definition ...]  [data calculation ...]  [variable definition ...]  [graph element ...]  [print element ...]  其中的 data definiton、variable definition 、data calculation、分别是下面的格式，  DEF:<vname>=<rrdfile>:<ds-name>:<CF>[:step=<step>][:start=<time>][:end=<time>][:reduce=<CF>]  VDEF:vname=RPN expression  CDEF:vname=RPN expression |

其中 filename 就是你想要生成的图片文件的名称，默认是 png 。你可以通过选项修改图片的类型，可以有 PNG、SVG、EPS、PDF四种。

(1).DEF 是 Definition （定义）的意思。定义什么呢？你要绘图，总要有数据源吧？DEF 就是告诉 RRDtool 从那个 RRD 中取出指定。

为什么还有一个 CF 字段？因为 RRA 有多种CF 类型，有些 RRA 可能用来保存平均值、有些 RRA 可能用于统计最大值、最小值等等。所以你必须同时指定使用什么 CF 类型的 RRA的数据。

至于 :start 和 :end 、:reduce 则用得比较少，最常用的就是 :step 了，它可以让你控制 RRDtool 从那个 RRA 中取数据。

(2).VDEF 是 Variable Definition （变量定义）的意思。定义什么呢？记得 MRTG 在图表的下面有一个称之为 Legend 的部分吗？那里显示了1个或者2个 DS （MRTG 没有 DS 一说，这里是借用 RRDtool 的）的 “最大值”、“平均值”、“当前值”。

RRDtool 中用 VDEF 来定义。这个变量专门存放某个 DS 某种类型的值，例如 eth0\_in 的最大值、eht0\_out 的当前值等。当你需要象MRTG 一样输出数字报表（Legend） 时，就可以在 GPRINT 子句（sub clause）中调用它。同样它也需要用一个变量来存放数值。要注意的是，旧版 的 RRDtool 中是用另外一种格式来达到相同的目的。新版的 RRDtool 则推荐使用VDEF语句。但在使用过程中，却发现 VDEF 的使用反而造成了困扰。 例如你有5个 DS 要画，每个 DS 你都想输出最大值、最小值、平均值 、当前值。 如果使用 VDEF ，则需要 4 \* 5 = 20 个 VDEF 语句，这会造成极大的困扰。具体例子可以看第十一节“数字报表”部分。

(3).CDEF 是 Calculation Define 的意思。使用过MRTG 的都会体会到一点，MRTG 的计算能力实在太差了。例如你有两个 Target ，一个是 eth0\_in ， 一个是 eth0\_out,如果要把它们相加起来，再除以8，得出 bytes 为单位的值，如何计算呢？或者说你只想看 eth0\_in 中超过 10Mb/s 的那部分， 其余的不关心，又如何实现呢？因为 MRTG 不能对它从 log 取出来的数据进行修改，只能原原本本的表现，所以很难满足我们的要求。而使用 CDEF ， 这一切都可以很容易的实现。CDEF 支持很多数学运算，甚至还支持简单的逻辑运算 if-then-else ，可以解决前面提到的第2个问题：如何只绘制你所关 心的数据。不过这一切都需要熟悉 RPN 的语法.所以我们放到下一节介绍，这一节就介绍把 RRDtool 中的数据以图表的方式显示出来。

(4).其它选项分类

本部分我们按照官方文档的方式，把选项分成几大类，分为 ：

* Time range ： 用于控制图表的X轴显示的起始/结束时间，也包括从RRA中提取指定时间的数据。
* Labels ：用于控制 X/Y 轴的说明文字。
* Size ：用于控制图片的大小。
* Limits ：用于控制 Y 轴的上下限。
* Grid ：用于控制 X/Y 轴的刻度如何显示。
* Miscellaneous ：其他选项。例如显示中文、水印效果等等。
* Report ：数字报表

注，需要说明的是，本博文中并不是列出了所有选项的用法，只是列出较为常用的选项，如果想查看所有选项的的用法，可以到官方站点下载文档。其实大部分选项我们都可以使用默认值不需要修改的。下面是常用选项，

|  |  |
| --- | --- |
| 1 | rrdtool graph filename [option ...] [data definition ...] [data calculation ...] [variable definition ...] [graph element ...] [print element ...] |

* filename 要绘制的图片名称
* Time range时间范围
* [-s|--start time] 启始时间[-e|--end time]结束时间 [-S|--step seconds]步长
* Labels
* [-t|--title string]图片的标题 [-v|--vertical-label string] Y轴说明
* Size
* [-w|--width pixels] 显示区的宽度[-h|--height pixels]显示区的高度 [-j|--only-graph]
* Limits
* [-u|--upper-limit value] Y轴正值高度[-l|--lower-limit value]Y轴负值高度 [-r|--rigid]
* Data and variables
* DEF:vname=rrdfile:ds-name:CF[:step=step][:start=time][:end=time]
* CDEF:vname=RPN expression
* VDEF:vname=RPN expression

好了，到这里我们RRDTool命令工具的基本使用，就讲解到这里了更多详细的内容请参考官方网站。好了，下面我们来演示一个完整的案例。

六、RRDTool绘图案例

案例：利用RRDTool来绘制mysql服务器查询次数的曲线图。

1.安装mysql服务器

|  |  |
| --- | --- |
| 1 | [root@node1 ~]# yum install -y mysql-server |

2.启动并测试

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | [root@node1 ~]# chkconfig mysqld on  [root@node1 ~]# service mysqld start  初始化 MySQL 数据库： Installing MySQL system tables...  OK  Filling help tables...  OK  To start mysqld at boot time you have to copy  support-files/mysql.server to the right place for your system  PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !  To do so, start the server, then issue the following commands:  /usr/bin/mysqladmin -u root password 'new-password'  /usr/bin/mysqladmin -u root -h node1.test.com password 'new-password'  Alternatively you can run:  /usr/bin/mysql\_secure\_installation  which will also give you the option of removing the test  databases and anonymous user created by default. This is  strongly recommended for production servers.  See the manual for more instructions.  You can start the MySQL daemon with:  cd /usr ; /usr/bin/mysqld\_safe &  You can test the MySQL daemon with mysql-test-run.pl  cd /usr/mysql-test ; perl mysql-test-run.pl  Please report any problems with the /usr/bin/mysqlbug script!                                [确定]  正在启动 mysqld：                     [确定]  [root@node1 ~]# mysql  Welcome to the MySQL monitor. Commands end with ; or \g.  Your MySQL connection id is 2  Server version: 5.1.69 Source distribution  Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.  Oracle is a registered trademark of Oracle Corporation and/or its  affiliates. Other names may be trademarks of their respective  owners.  Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  mysql> show databases;  +--------------------+  | Database      |  +--------------------+  | information\_schema |  | mysql       |  | test        |  +--------------------+  3 rows in set (0.00 sec)  mysql> |

3.创建RRD数据库文件

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | [root@node1 ~]# rrdtool create mysql.rrd --step 3 DS:mysqlselect:COUNTER:5:0:U RRA:AVERAGE:0.5:1:28800 RRA:AVERAGE:0.5:10:2880 RRA:MAX:0.5:10:2880 RRA:LAST:0.5:10:2880  [root@node1 ~]# ll -h  总用量 620K  -rw-------. 1 root root 970 8月 17 18:50 anaconda-ks.cfg  -rw-r--r--. 1 root root 16K 8月 17 18:50 install.log  -rw-r--r--. 1 root root 4.1K 8月 17 18:48 install.log.syslog  -rw-r--r-- 1 root root 294K 10月 11 15:57 mysql.rrd  -rw-r--r-- 1 root root 294K 10月 10 21:53 rrdtool-1.3.8-6.el6.x86\_64.rpm |

4.抓取数据

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# mysql --batch -e "show global status like 'com\_select'" | awk '/Com\_select/{print $2}'  5 |

5.更新RRD数据库

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | [root@node1 ~]# vim getselect.sh  #!/bin/bash  #  while true; do      SELECT=`mysql --batch -e "show global status like 'com\_select'" | awk '/Com\_select/{print $2}'`      rrdtool update mysql.rrd N:$SELECT      sleep 3  done  [root@node1 ~]# bash -n getselect.sh  [root@node1 ~]# bash -x getselect.sh  + true  ++ mysql --batch -e 'show global status like '\''com\_select'\'''  ++ awk '/Com\_select/{print $2}'  + SELECT=10  + rrdtool update mysql.rrd N:10  + sleep 3  + true  ++ awk '/Com\_select/{print $2}'  ++ mysql --batch -e 'show global status like '\''com\_select'\'''  + SELECT=11  + rrdtool update mysql.rrd N:11  + sleep 3  + true  ++ mysql --batch -e 'show global status like '\''com\_select'\'''  ++ awk '/Com\_select/{print $2}'  + SELECT=12  + rrdtool update mysql.rrd N:12 |

注，让这个脚本一直执行着。

6.创建一个测试数据库

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | [root@node1 ~]# mysql  Welcome to the MySQL monitor. Commands end with ; or \g.  Your MySQL connection id is 55  Server version: 5.1.69 Source distribution  Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.  Oracle is a registered trademark of Oracle Corporation and/or its  affiliates. Other names may be trademarks of their respective  owners.  Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  mysql> create database testdb;  Query OK, 1 row affected (0.01 sec)  mysql> use testdb;  Database changed  mysql> create table tb1 (id int UNSIGNED NOT NULL AUTO\_INCREMENT PRIMARY KEY, name char(50) NOT NULL);  Query OK, 0 rows affected (0.02 sec)  mysql> show tables;  +------------------+  | Tables\_in\_testdb |  +------------------+  | tb1       |  +------------------+  1 row in set (0.00 sec)  mysql> |

7.创建一个脚本不断的插入数据并查询

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | [root@node1 ~]# vim insert.sh  #!/bin/bash  #  for I in {1..200000}; do      mysql -e "INSERT INTO testdb.tb1(name) VALUES ('stu$I')"      mysql -e "SELECT \* FROM testdb.tb1" &> /dev/null  done  ~  [root@node1 ~]# bash -n insert.sh  [root@node1 ~]# bash -x insert.sh |

8.查看一下RRD数据库更新的数据

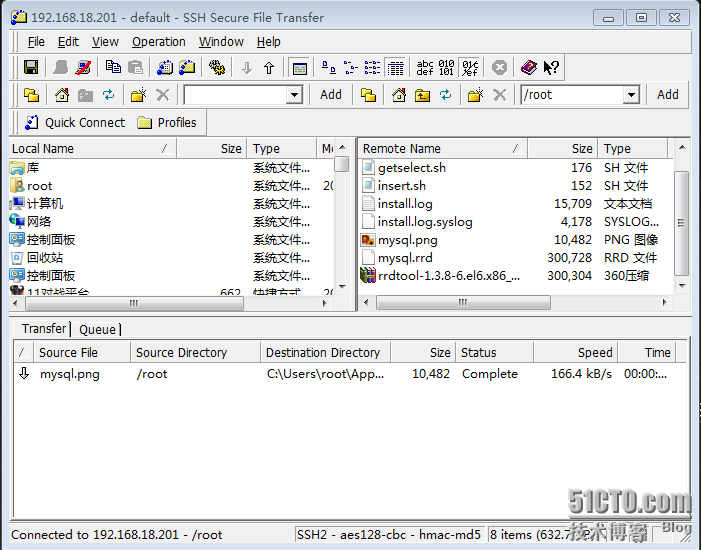
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79 | [root@node1 ~]# rrdtool fetch -r 3 mysql.rrd AVERAGE  1381478757: -nan  1381478760: -nan  1381478763: -nan  1381478766: -nan  1381478769: -nan  1381478772: -nan  1381478775: 2.7153386392e-01  1381478778: 3.2831536999e-01  1381478781: 3.2891623754e-01  1381478784: 3.2705226490e-01  1381478787: 3.2799497906e-01  1381478790: 3.2750147283e-01  1381478793: 3.2962107218e-01  1381478796: 3.3022497969e-01  1381478799: 3.3027211905e-01  1381478802: 3.3020369194e-01  1381478805: 3.2946024073e-01  1381478808: 3.2988230260e-01  1381478811: 3.2969005472e-01  1381478814: 3.2974230463e-01  1381478817: 3.3001057711e-01  1381478820: 3.3019278582e-01  1381478823: 3.3083777490e-01  1381478826: 3.3015850009e-01  1381478829: 3.2968813815e-01  1381478832: 3.3021007195e-01  1381478835: 3.2890877932e-01  1381478838: 3.2919982365e-01  1381478841: 3.2820752812e-01  1381478844: 3.2498916047e-01  1381478847: 3.2435105446e-01  1381478850: 3.2631508451e-01  1381478853: 3.2927988387e-01  1381478856: 3.3061808059e-01  1381478859: 3.3065099981e-01  1381478862: 3.3079060547e-01  1381478865: 3.2993297013e-01  1381478868: 3.2998088978e-01  1381478871: 3.3045720109e-01  1381478874: 3.3052361682e-01  1381478877: 3.3021445518e-01  1381478880: 3.3033678729e-01  1381478883: 3.3017146110e-01  1381478886: 3.2932443118e-01  1381478889: 3.2872916025e-01  1381478892: 3.2942230122e-01  1381478895: 3.3004157568e-01  1381478898: 3.3035752652e-01  1381478901: 3.3026495130e-01  1381478904: 4.2927608935e-01  1381478907: 5.6199888336e-01  1381478910: 3.2960053815e-01  1381478913: 3.3019513627e-01  1381478916: 3.3008973582e-01  1381478919: 3.3023471404e-01  1381478922: 3.3044897038e-01  1381478925: 3.3025127245e-01  1381478928: 3.2999671137e-01  1381478931: 3.2995130475e-01  1381478934: 3.3001845566e-01  1381478937: 3.3004261932e-01  1381478940: 3.2985954162e-01  1381478943: 3.2962262303e-01  1381478946: 3.3033462847e-01  1381478949: 3.3000997317e-01  1381478952: 3.3023836505e-01  1381478955: 3.2987551061e-01  1381478958: 3.3038940726e-01  1381478961: 3.3047901095e-01  1381478964: 3.2999606597e-01  1381478967: 3.3021352982e-01  1381478970: 3.2998445954e-01  1381478973: 3.3029458891e-01  1381478976: 3.3009257605e-01  1381478979: 3.3008453893e-01  1381478982: 3.2998650516e-01  1381478985: 3.3014434356e-01  1381478988: 3.2950044395e-01 |

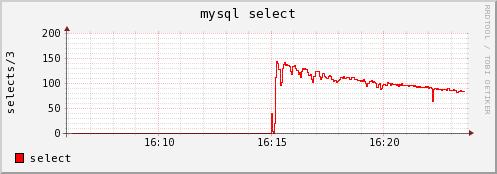
注，大家可以看到现在已经有很多的数据了，下面我们来简单的制作一查询曲线图。

9.制作查询曲线图

|  |  |
| --- | --- |
| 1  2 | [root@node1 ~]# rrdtool graph mysql.png -s 1381478754 -t "mysql select" -v "selects/3" DEF:select3=mysql.rrd:mysqlselect:AVERAGE:step=3 LINE1:select3#FF0000:"select"  497x174 |

10.取出mysql.png图片并查看

[](http://img1.51cto.com/attachment/201310/11/2033581_1381480512gZBX.png)

[](http://img1.51cto.com/attachment/201310/11/2033581_1381480518MMhs.png)

好了，到这里我们RRDTool的讲解就全部结束了，想了解更多的RRDTool相关的知识一方面大家可以参考官方文档，另一方面大家可以参考一下这篇博客<http://bbs.chinaunix.net/forum.php?mod=viewthread&tid=864861&page=1>，我认为还是写的比较详细的，好了这一篇博客就到这边了，希望大家有所收获吧^\_^……

本文出自 “[Share your knowledge …](http://freeloda.blog.51cto.com/)” 博客，请务必保留此出处<http://freeloda.blog.51cto.com/2033581/1307492>

## Sender

### @配置

{

"debug": true,

"http": {

"enabled": true,

"listen": "0.0.0.0:6066"

},

"redis": {

"addr": "127.0.0.1:6379",

"maxIdle": 5

},

"queue": {

"sms": "/sms",

"mail": "/mail"

},

"worker": {

"sms": 10,

"mail": 50

},

"api": {

"sms": "http://11.11.11.11:8000/sms",

"mail": "http://192.168.31.61:8080/monitor/email"

}

}

"mail": "http://192.168.31.220:8180/monitor/email"

### sender\cron\mail.go SendMail发送邮件

E:\workspace\yh\OpenBridge-passos-proxy\open-faclon\src\

sender\cron\mail.go

**func** SendMail(mail \*model.Mail) {  
 **defer func**() {  
 <-MailWorkerChan  
 }()  
  
 url := g.Config().Api.Mail  
 r := httplib.Post(url).SetTimeout(5\*time.*Second*, 2\*time.*Minute*)  
 r.Param("tos", mail.Tos)  
 r.Param("subject", mail.Subject)  
 r.Param("content", mail.Content)  
 resp, err := r.String()  
 **if** err != nil {  
 log.Println(err)  
 }  
  
 proc.IncreMailCount()  
  
 **if** g.Config().Debug {  
 log.Println("==mail==>>>>", mail)  
 log.Println("<<<<==mail==", resp)  
 }  
  
}

### redis\pop.go PopAllMail发送邮件

PopAllMail

E:\workspace\yh\OpenBridge-passos-proxy\open-faclon\src\

github.com\open-falcon\sender\redis\pop.go

### Sender实例

2016/08/09 15:31:34 cfg.go:81: read config file: cfg.json successfully

2016/08/09 15:31:34 http.go:60: http listening 0.0.0.0:6066

2016/08/09 17:09:20 mail.go:44: Post http://11.11.11.11:9000/mail: dial tcp 11.11.11.11:9000: i/o timeout

2016/08/09 17:09:20 mail.go:50: ==mail==>>>> <Tos:reymont@sina.cn,, Subject:[P3][PROBLEM][192.168.1.55][][ all(#3) cpu.busy 1>=90][O0 1970-01-01 08:00:00], Content:PROBLEM

P3

Endpoint:192.168.1.55

Metric:cpu.busy

Tags:

all(#3): 1>=90

Note:

Max:3, Current:0

Timestamp:1970-01-01 08:00:00

http://127.0.0.1:5050/template/view/7

>

2016/08/09 17:09:20 mail.go:51: <<<<==mail==

2016/08/09 17:12:22 sms.go:43: Post http://11.11.11.11:8000/sms: dial tcp 11.11.11.11:8000: i/o timeout

2016/08/09 17:12:22 sms.go:49: ==sms==>>>> <Tos:, Content:[P1][PROBLEM][192.168.1.138][][ all(#3) cpu.busy 4>=2][O3 2016-08-09 17:12:00]>

2016/08/09 17:12:22 sms.go:50: <<<<==sms==

2016/08/09 17:12:22 mail.go:44: Post http://11.11.11.11:9000/mail: dial tcp 11.11.11.11:9000: i/o timeout

2016/08/09 17:12:22 mail.go:50: ==mail==>>>> <Tos:reymont@sina.cn,, Subject:[P1][PROBLEM][192.168.1.138][][ all(#3) cpu.busy 4>=2][O3 2016-08-09 17:12:00], Content:PROBLEM

P1

Endpoint:192.168.1.138

Metric:cpu.busy

Tags:

all(#3): 4>=2

Note:

Max:3, Current:3

Timestamp:2016-08-09 17:12:00

http://127.0.0.1:5050/template/view/7

>

2016/08/09 17:12:22 mail.go:51: <<<<==mail==

2016/08/09 17:13:23 mail.go:44: Post http://11.11.11.11:9000/mail: dial tcp 11.11.11.11:9000: i/o timeout

2016/08/09 17:13:23 mail.go:50: ==mail==>>>> <Tos:reymont@sina.cn,, Subject:[P1][PROBLEM][192.168.1.55][][ all(#3) cpu.busy 4.0404>=2][O2 2016-08-09 17:13:00], Content:PROBLEM

P1

Endpoint:192.168.1.55

Metric:cpu.busy

Tags:

all(#3): 4.0404>=2

Note:

Max:3, Current:2

Timestamp:2016-08-09 17:13:00

http://127.0.0.1:5050/template/view/7

>

2016/08/09 17:13:23 mail.go:51: <<<<==mail==

2016/08/09 17:13:23 sms.go:43: Post http://11.11.11.11:8000/sms: dial tcp 11.11.11.11:8000: i/o timeout

2016/08/09 17:13:23 sms.go:49: ==sms==>>>> <Tos:, Content:[P1][PROBLEM][192.168.1.55][][ all(#3) cpu.busy 4.0404>=2][O2 2016-08-09 17:13:00]>

2016/08/09 17:13:23 sms.go:50: <<<<==sms==

### 一封邮件里发送两个Expression

<Tos:ymy@qq.com, Subject:[P6][PROBLEM] 2 container.cpu.usage.busy, Content:PROBLEM^M

P6^M

Endpoint:192.168.1.78^M

Metric:container.cpu.usage.busy^M

Tags:deploy\_id=71m4i2p94jneidpeffvkpny4crewuuo,id=3ea72a9d4d1d7060f0d5f78d706a2576fa6f968743cfc6621b1deb862f174a11^M

avg(#3): 0.03357>=0.01^M

Note:^M

Max:3, Current:3^M

Timestamp:2016-10-19 01:39:00^M

https://test.yihecloud.com/monitor/expression/view/268^M

^M

PROBLEM^M

P6^M

Endpoint:192.168.1.79^M

Metric:container.cpu.usage.busy^M

Tags:deploy\_id=70pf4ifo1pmj6zvcefjbaurbsemgbvi,id=ac0c9cd4f0d98e653015bde845355639dd444e3cc941f11b75071edf8e771a75^M

avg(#3): 0.05078>=0.01^M

Note:^M

Max:3, Current:3^M

Timestamp:2016-10-19 01:39:00^M

https://test.yihecloud.com/monitor/expression/view/253^M

>

## Mail-provider

### open-falcon/mail-provider

https://github.com/open-falcon/mail-provider

curl http://$ip:4000/sender/mail -d "tos=a@a.com,b@b.com&subject=xx&content=yy"

### 邮件短信发送接口 | Open-Falcon

http://book.open-falcon.org/zh/install\_from\_src/mail-sms.html

这个组件没有代码，需要各个公司自行提供。

监控系统产生报警事件之后需要发送报警邮件或者报警短信，各个公司可能有自己的邮件服务器，有自己的邮件发送方法；有自己的短信通道，有自己的短信发送方法。falcon为了适配各个公司，在接入方案上做了一个规范，需要各公司提供http的短信和邮件发送接口

短信发送http接口：

method: post

params:

- content: 短信内容

- tos: 使用逗号分隔的多个手机号

邮件发送http接口：

method: post

params:

- content: 邮件内容

- subject: 邮件标题

- tos: 使用逗号分隔的多个邮件地址

## FAQ

### dashboard 空白 errno: 0x023a, str:opening error · Issue #12 · open-falcon/graph

https://github.com/open-falcon/graph/issues/12

[root@fdc055c](https://github.com/root/graph/commit/fdc055cabfe3):/home/work/open-falcon# cat logs/graph.log  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error  
errno: 0x023a, str:opening error

使用docker镜像默认配置部署的，dashboard 全部是空白没数据，发现这个错误信息, 不明白什么意思。

"rrd": {  
"storage": "/home/work/open-falcon/data/6070"  
},  
这个目录下是空的，感觉是写数据打开文件失败？，但root启动有写入权限的

这个错误日志 是说磁盘忙、写入数据有少量失败，不是没有写权限，也不会导致dashboard空白。 按照book里的faq排查下吧。

### SMART

Hard drives use S.M.A.R.T. (Self-Monitoring, Analysis, and Reporting Technology) to gauge([geɪdʒ]估计) their own reliability and determine if they’re failing. You can view your hard drive’s S.M.A.R.T. data and see if it has started to develop problems.

S.M.A.R.T. - Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/S.M.A.R.T.

S.M.A.R.T. (Self-Monitoring, Analysis and Reporting Technology; often written as SMART) is a monitoring system included in computer hard disk drives (HDDs) and solid-state drives (SSDs)[1] that detects and reports on various indicators of drive reliability, with the intent of enabling the anticipation of hardware failures.

When S.M.A.R.T. data indicates a possible imminent(['ɪmɪnənt] 即将来临的;) drive failure, software running on the host system may notify the user so stored data can be copied to another storage device, preventing data loss, and the failing drive can be replaced.

### 磁盘SMART信息详解 - cgm88s的专栏 - 博客频道 - CSDN.NET

http://blog.csdn.net/chengm8/article/details/28601097

1,机械硬盘的SMART表定义已经有自己的标准，由于硬盘厂很多，很多厂家属性的名字也不尽相同，

或者某些厂牌缺少某些属性，但是同个ID的定义是相同的。机械硬盘的SMART属性表如下：

| ID | ID十六进制值 | 英文名 | 中文译名 | 最优 | 说明 |
| --- | --- | --- | --- | --- | --- |
| 1 | 0x01 | read error rate | 底层数据读取错误率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 存储器从一个硬盘表面读取数据时发生的错误率。原始值由于不同厂商的不同计算方法而有所不同，其十进制值往往无意义的。一般来说有数值意味着磁头已出现问题了。 |
| 2 | 0x02 | Throughput Performance | 读写通量性能 | [Dark Green Arrow Up.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Up.svg) | 通常是硬盘读写性能的测量值，如果其值有变动，有可能硬盘出现了问题。 |
| 3 | 0x03 | Spin-Up Time | 盘片启动时间 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 盘片由静止启动加速到稳定正常运行速度的平均所需时间。 |
| 4 | 0x04 | Start/Stop Count | 电机起停次计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 一个盘片启动关闭周期的统计值，只有硬盘从完全断电中启动或从睡眠模式恢复，盘片主轴电机被启动时才会记一次数。 |
| 5 | 0x05 | Reallocated Sector Count | 重定位磁区计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录由于损坏而被映射到无损的后备区的扇区计数。当硬盘出现损坏扇区时，可以通过将其物理空间指向到特定的无损区域进行重映射修复，从而出现坏扇区的硬盘仍可使用。但当高过一定数值后，后扇区消耗殆尽而无法再重映射修复时，这些坏扇区就会显现出来且无法自行修复。除外由于要要求磁头读取这些坏扇区时专门再移动到后备区读写数据，对硬盘读写性能也有影响。 |
| 6 | 0x06 | Read Channel Margin | 信道读取余量 |  | 读取数据时信道可用的余量，该属性没制定任何功用。 |
| 7 | 0x07 | Seek Error Rate | 寻道错误率 |  | （该属性是特定制造商才有的）磁头寻找磁道由于机械问题而出错几率，有多种原因可能引致出错，如：磁头伺服构件，盘体过热，或损坏。于不同厂商的不同计算方法而有所不同，其十进制值往往无意义的。 |
| 8 | 0x08 | Seek Time Performance | 寻道性能 | [Dark Green Arrow Up.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Up.svg) | 每次寻道时间的平均值，该值短期内迅速减少，有可能硬盘出现了问题。 |
| 9 | 0x09 | Power-On Hours | 硬盘加电时间 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 硬盘自出厂以来加电启动的统计时间，单位为小时（或根据制造商设定为分钟或秒），一般用户以该值判定硬盘是否被使用过。 |
| 10 | 0x0a | Spin Retry Count | 电机起转重试 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | S.M.A.R.T参数电机起转重试，表明了主轴电机的启动尝试次数。这个属性存储了关于主轴电机尝试加速到完全可操作速度的次数（在这种情况下，意味着主轴电机的第一次启动尝试没有成功）。主轴电机频繁的尝试启动，意味着硬盘驱动器的寿命可能将近实际限值。 |
| 11 | 0x0b | Recalibration Retries | 磁头校准重试 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 磁头在一次运行失败时尝试校准至正常状态的统计数，该值改变时意味着硬盘的机械部件已经出现问题了。 |
| 12 | 0x0c | Power Cycle Count | 设备开关计数 |  | 该属性表示硬盘电源充分开/关循环计数。 |
| 13 | 0x0d | Soft Read Error Rate | 软件读取错误率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 操作系统读取数据时的出错率。 |
| 183 | 0xb7 | SATA Downshift Error Count | SATA降级运行计数 |  | Western Digital 和 Samsung 特有属性，记录由于兼容问题导致降低SATA传输级别运行的计数。 |
| 184 | 0xb8 | End-to-End error | 终端校验出错 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | HP专有S.M.A.R.T.（SMART IV）技术的一个特有属性，记录硬盘从盘片读取数据到高速缓存后再传输到主机时数据校验出错的次数。 |
| 185 | 0xb9 | Head Stability | 磁头稳定性 |  | Western Digital特有属性 |
| 186 | 0xba | Induced Op-Vibration Detection |  |  | Western Digital特有属性 |
| 187 | 0xbb | Reported Uncorrectable Errors | 报告不可纠正错误 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 硬件ECC无法恢复的错误计数。 |
| 188 | 0xbc | Command Timeout | 通信超时 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 由于无法连接至硬盘而终止操作的统计数，一般为0，如果远超过0，则可能电源问题，数据线接口氧化或更严重的问题。 |
| 189 | 0xbd | High Fly Writes | 磁头写入高度 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 硬盘进行写入时对磁头高度进行监控以提供额外的保障。当磁头处于不正常高度进行写入时，写入操作会被终止，原有数据重写入或者将该扇区重映射到安全区域。该属性是统计值。 |
| 190 | 0xbe | Airflow Temperature | 气流温度 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | Western Digital特有属性，计量硬盘内气流温度，和检测项0xc2相似。 |
| 191 | 0xbf | G-sense Error Rate | 加速度错误率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 计量可能对硬盘做成损害的冲击次数。 |
| 192 | 0xc0 | Power-off Retract Count | 电源关闭磁头收回计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 计量磁头在没有加电时不移进硬盘的值。 |
| 193 | 0xc1 | Load Cycle Count | 磁头升降计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 计量磁头在加电时移进/移出硬盘周期的值。 |
| 194 | 0xc2 | Temperature | 温度 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 计量硬盘的温度 |
| 195 | 0xc3 | Hardware ECC Recovered | 硬件ECC恢复 |  | （特定原始值） |
| 196 | 0xc4 | Reallocation Event Count | 重定位事件计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录已重映射扇区和可能重映射扇区的事件计数。 |
| 197 | 0xc5 | Current Pending Sector Count | 等候重定的扇区计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录了不稳定的扇区的数量。 |
| 198 | 0xc6 | Uncorrectable Sector Count | 无法校正的扇区计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录肯定出错的扇区数量。 |
| 199 | 0xc7 | UltraDMA CRC Error Count | UltraDMA通讯CRC错误 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录硬盘通讯时发生的CRC错误。 |
| 200 | 0xc8 | Multi-Zone Error Rate | 多区域错误率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 写入一个区域时发现的错误的计数。 |
| 200 | 0xc8 | Write Error Rate | 写入错误率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | Fujitsu的特别属性，写入一个区域时发现的错误的计数。 |
| 201 | 0xc9 | Soft Read Error Rate | 逻辑读取错误率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录脱轨错误。 |
| 202 | 0xca | Data Address Mark errors | 数据地址标记错误 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录数据地址标记错误（或制造商特定的计数） |
| 203 | 0xcb | Run Out Cancel | 用完取消 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | ECC错误计数 |
| 204 | 0xcc | Soft ECC Correction | 逻辑ECC纠正 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录由软件ECC更正的错误计数。 |
| 205 | 0xcd | Thermal Asperity Rate | 热嘈率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录高温导致的出错记数。 |
| 206 | 0xce | Flying Height | 飞行高度 |  | 记录磁头的飞行高度。飞得太低会增加磁头撞毁的机会，飞得太高增加读写错误的机会。 |
| 207 | 0xcf | Spin High Current | 主轴电机浪涌电流计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录主轴电机运转时浪涌电流的次数。 |
| 208 | 0xd0 | Spin Buzz |  |  | 记录由于电力不足而启动主轴电机的蜂鸣声次数。 |
| 209 | 0xd1 | Offline Seek Performance | 离线寻址性能 |  | 在其内部测试硬盘的寻址能力表现。 |
| 210 | 0xd2 | ？ | ？ |  | （没定性，出现在Maxtor 6B200M0 200GB 和Maxtor 2R015H1 15GB 的硬盘中） |
| 211 | 0xd3 | Vibration During Write | 写操作震动 |  | 记录写入操作的震动数。 |
| 212 | 0xd4 | Shock During Write | 写操作冲击 |  | 记录写入操作时的冲击数。 |
| 220 | 0xdc | Disk Shift | 盘体偏移 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录盘体由于冲击或温度导致偏离主轴的相对距离。 |
| 221 | 0xdd | G-Sense Error Rate | 加速计出错率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 从外部诱发的冲击和振动产生的错误计数。 |
| 222 | 0xde | Loaded Hours | 数据加载时间 |  | 数据读取时所花费的时间。（磁头移动时间） |
| 223 | 0xdf | Load/Unload Retry Count | 加载/卸载重试次数 |  | 磁头改变位置时所需时间。 |
| 224 | 0xe0 | Load Friction | 负载摩擦 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 读写时由于机械摩擦做成的阻力。 |
| 225 | 0xe1 | Load/Unload Cycle Count | 加载/卸载循环计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 总负载周期计数。 |
| 226 | 0xe2 | Load 'In'-time | 磁头 |  | 磁头加载所需总时间（不包括在停泊区的花费）。 |
| 227 | 0xe3 | Torque Amplification Count | 扭矩放大计数 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 尝试来补偿盘片的速度变化的计数。 |
| 228 | 0xe4 | Power-Off Retract Cycle | 断电缩回周期 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 切断电源后电磁枢自动缩回的时间计数。 |
| 230 | 0xe6 | GMR Head Amplitude | GMR磁头振幅 |  | 磁头振幅计数（磁头反复正反向运动距离）。 |
| 231 | 0xe7 | Temperature | 硬盘温度 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 记录硬盘温度。 |
| 232 | 0xe8 | Endurance Remaining | 耐久性剩余 |  | 磁盘可使用周期与设计可使用周期的百分比。 |
| 232 | 0xe8 | Available Reserved Space | 可用保留空间 |  | Intel固态硬盘报告的可提供的预留空间占作为一支全新的固态硬盘预留空间的百分比。 |
| 233 | 0xe9 | Power-On Hours | 加电时间 |  | 处于开机状态的小时数。 |
| 233 | 0xe9 | Media Wearout Indicator | 介质耗损指标 |  | Intel固态硬盘报告的NAND刷写寿命，全新时值为100，最低值为1，其跌幅随NAND的擦除周期增加而在0到最大额定周期范围减少。 |
| 240 | 0xf0 | Head Flying Hours | 磁头飞行时间 |  | 磁头处于定位中的时间。 |
| 240 | 0xf0 | Transfer Error Rate | 传输错误率 |  | 在数据传输时连接被重置的次数计数。（Fujitsu特有属性） |
| 241 | 0xf1 | Total LBAs Written | LBA写入总数 |  | LBA写入总数计数。 |
| 242 | 0xf2 | Total LBAs Read | LBA读取总数 |  | LBA读取总数计数，部分S.M.A.R.T.检测程序会把原始值显示为负数，这是因为该原始值为48位，而不是32位的。 |
| 250 | 0xfa | Read Error Retry Rate | 读取错误重试率 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 从磁盘读取时的错误计数。 |
| 254 | 0xfe | Free Fall Protection | 自由跌落保护 | [Dark Green Arrow Down.svg](http://zh.wikipedia.org/wiki/File:Dark_Green_Arrow_Down.svg) | 对“自由落体事件”检测计数。 |

SMART属性解释:

1，ID# :  属性ID, 从1到255.

2，ATTRIBUTE\_NAME : 属性名.

3，FLAG : 表示这个属性携带的标记. 使用-f brief可以打印.

4，VALUE: Normalized value正常值, 取值范围1到254. 越低表示越差. 越高表示越好.

当前值是各ID项在硬盘运行时根据实测数据(RAW\_VALUE)通过公式计算的结果，计算公式由硬盘厂家自定。 硬盘出厂时各ID项目都有一个预设的最大正常值，也即出厂值，这个预设的依据及计算方法为硬盘厂家保密，不同型号的硬盘都不同，最大正常值通常为100或200或253，

新硬盘刚开始使用时显示的当前值可以认为是预设的最大正常值（有些ID项如温度等除外）。

随着使用损耗或出现错误，当前值会根据实测数据而不断刷新并逐渐减小。

因此，当前值接近临界值就意味着硬盘寿命的减少，发生故障的可能性增大，所以当前值也是判定硬盘健康状态或推测寿命的依据之一。

5,WORST: 最差值，表示SMART开启以来的, 所有Normalized values的最低值。

最差值是硬盘运行时各ID项曾出现过的最大的非正常值。

最差值是对硬盘运行中某项数据变劣的峰值统计，该数值也会不断刷新。

通常，最差值与当前值是相等的，如果最差值出现较大的波动（小于当前值），表明硬盘曾出现错误或曾经历过恶劣的工作环境（如温度）。

6，THRESH：阈值。当Normalized value小于等于THRESH值时, 表示这项指标已经failed了。

注意, 如果这个属性是pre-failure的, 那么这项如果出现Normalized value<=THRESH, 那么磁盘将马上failed掉.

7，TYPE:这里存在两种TYPE类型, Pre-failed和Old\_age.

Pre-failed 类型的Normalized value可以用来预先知道磁盘是否要坏了. 例如Normalized value接近THRESH时, 就赶紧换硬盘吧.

Old\_age 类型的Normalized value是指正常的使用损耗值, 当Normalized value 接近THRESH时, 也需要注意, 但是比Pre-failed要好一点.

8,UPDATED:这个字段表示这个属性的值在什么情况下会被更新.

一种是通常的操作和离线测试都更新(Always),

另一种是只在离线测试的情况下更新(Offline).

9,WHEN\_FAILED:这个字段表示当前这个属性的状态。取值有以下三种：

failing\_now(normalized\_value <= THRESH),

或者in\_the\_past(WORST <= THRESH),

或者 - , 正常(normalized\_value以及wrost >= THRESH).

10，RAW\_VALUE:表示这个属性的未转换前的RAW值, 可能是计数, 也可能是温度, 也可能是其他的.

注意RAW\_VALUE转换成Normalized value是由厂商的firmware提供的, smartmontools不提供转换.

2,固态硬盘(SSD)的SMART表定义则目前还没有统一标准，不同厂家甚至不同主控都有可能出现相同ID不同定义，  
所以用一般的SMART软件查看是没任何意义的，虽然你可以看到值，但是这个值对应的ID解释可能完全不是那么回事。

不同主控SSD的SMART属性有：

 intel SSD SMART:

03 – Spin Up Time （磁头加载时间）

04 – Start/Stop Count （开始/停止计数）

05 – Re-Allocated Sector Count （重映射扇区数）

09 – Power-On Hours Count （通电时间）

0C – Power Cycle Count （通断电次数）

C0 – Unsafe Shutdown Count （异常关机次数）

E1 – Host Writes （数据写入量）

E8 – Available Reserved Space （可用预留空间）（这个算是颗粒寿命，等于低于10%SSD就离躺倒不远了）

E9 – Media Wearout Indicator （闪存磨耗指数）

B8 – End to End Error Detection Count （端对端错误监测数）

SandForce SSD SMART:  
1-Raw Read Error Rate   底层数据读取出错率  
5-Retired Block Count 不可使用的坏块计数 （公式比较怪。。这个值不准，新固件都为100）  
9-Power On Hours Count    累计加电时间  
12-Power Cycle Count   设备通电周期  
171-Program Fail Count       编程错误计数  
172-Erase Fail Count         擦除错误计数  
174-Unexpected Power Loss Count    不正常掉电次数  
177-Wear-Range Data 显示最大磨损块和最小磨损块相差的百分比  
181-同171定义相同  
182-同172定义相同  
187-Reported Uncorrectable Errors 不可修复错误计数  
194-显示温度的，基本可以忽略（假的）  
195-On the Fly Reported Uncorrectable Error Count    实时不可修复错误计数  
196-Reallocated Event Count                      重映射坏块计数  
231-SSD Life left      SSD剩余寿命   
     新盘为100，当显示为10，代表P/E用完了，但是还有备用空间可以替换，显示0代表盘上数据为只读。  
241-lifetime write froms host         来自主机的写入数据量总数（64G更新一次）  
242-lifetime write froms host         来自主机的读取数据量总数（64G更新一次）

Micron(镁光)SSD SMART:  
1-Raw Read Error Rate                          底层数据读取出错率  
5-Re-allocated Sectors Count                 使用中新增的坏块数  
9-Power On Hours Count                       累计加电时间  
12-Power Cycle Count                             设备通电周期  
170-Grown Failing Block Count                   替换坏块计数  
171-Program Fail Count                             编程错误计数  
172-Erase Fail Count                                 擦除错误计数  
173-Wear Leveling Count                          平均擦写次数  
174-Unexpected Power Loss Count            不正常掉电次数  
181-Non-4k Aligned Access                       非4KB对齐访问数  
183-SATA Interface Downshift                   接口降级次数计数  
187-Reported Uncorrectable Errors            不可修复错误计数  
188-Command Timeout                            指令超时计数  
189-Factory Bad Block Count                    出厂坏块计数  
196-Re-allocation Event Count                  坏块重映射事件计数  
197-Current Pending Sector Count           值永远为0  
198-Smart Off-line Scan Uncorrectable Error Count     自检时发现的不可修复错误  
199-Ultra DMA CRC Error Rate                 主机到接口之间传输CRC错误率  
202-Percentage Of The Rated Lifetime Used   剩余寿命（MLC 5000 / SLC 100000计算)   
                       百分比从100开始跌  
206-Write Error Rate 底层数据写入出错率  
我觉得最主要的是那个173/AD的值，那个值是平均块擦写次数，用户可以靠它判断自己的盘剩余寿命。

Indilinx SSD SMART:  
1-Raw Read Error Rate底层数据读取出错率  
9-Power On Hours Count累计加电时间  
12-Power Cycle Count设备通电周期  
184-Init Bad Block Count坏块数  
195-Program Failure block Count编程错误块计数  
196-Erase Failure block Count擦除错误块计数  
197-Read Failure block Count读取错误块计数（不可修复错误）  
198-Total Count of Read Sectors总读取页数  
199-Total Count of Write Sectors总写入页数  
200-Total Count of Read Command总读取指令数  
200-Total Count of Write Command总写入指令数  
202-Total Count of error bits from flash总闪存错误bit数  
203-Total Count of Read Sectors with correct bits error  总修复bit错误的读取页数字  
204-BAD Block Full Flag  
205-Max P/E Count最大可编程/擦除次数  MLC 5000/10000   or SLC 100000  
206-Erase Count Min最小擦写次数  
207-Erase Count Max最大擦写次数  
208-Erase Count Average平均擦写次数  
209-Remaining Life %剩余寿命百分比  
210-BBM Error Log坏块管理错误日志  
211-SATA Error Count CRC (Write)     SATA 主机 <->接口CRC写入错误计数  
212-SATA Error Count HANDSHAKE (Read)  SATA 主机 <->接口读取错误计数

### 采集相关 | Open-Falcon

http://book.open-falcon.org/zh/faq/collect.html

curl http://127.0.0.1:6081/history/$endpoint/$counter

curl http://127.0.0.1:6081/history/localhost /cup.idle

curl -s http://127.0.0.1:6071/history/localhost/agent.alive

http://192.168.1.135:6071/history/192.168.1.70/agent.alive

http://192.168.1.136:1988/health

数据收集相关问题

Open-Falcon数据收集，分为[绘图数据]收集和[报警数据]收集。下面介绍，如何验证两个链路的数据收集是否正常。

如何验证[绘图数据]收集是否正常

数据链路是：agent->transfer->graph->query->dashboard。graph有一个http接口可以验证agent->transfer->graph这条链路，比如graph的http端口是6071，可以这么访问验证：

# $endpoint和$counter是变量

curl http://127.0.0.1:6071/history/$endpoint/$counter

# 如果上报的数据不带tags，访问方式是这样的:

curl http://127.0.0.1:6071/history/host01/agent.alive

# 如果上报的数据带有tags，访问方式如下，其中tags为module=graph,project=falcon

curl http://127.0.0.1:6071/history/host01/qps/module=graph,project=falcon

如果调用上述接口返回空值，则说明agent没有上报数据、或者transfer服务异常。

如何验证[报警数据]收集是否正常

数据链路是：agent->transfer->judge，judge有一个http接口可以验证agent->transfer->judge这条链路，比如judge的http端口是6081，可以这么访问验证：

curl http://127.0.0.1:6081/history/$endpoint/$counter

# $endpoint和$counter是变量，举个例子：

curl http://127.0.0.1:6081/history/host01/cpu.idle

# counter=$metric/sorted($tags)

# 如果上报的数据带有tag，访问方式是这样的，比如：

curl http://127.0.0.1:6081/history/host01/qps/module=judge,project=falcon

如果调用上述接口返回空值，则说明agent没有上报数据、或者transfer服务异常。