

Collectd

一.简介

collectd是一个守护(daemon)进程，用来定期收集系统和应用程序的性能指标，同时提供了以不同的方式来存储这些指标值的机制。

collectd从各种来源收集指标，例如 操作系统，应用程序，日志文件和外部设备，并存储此信息或通过网络使其可用。这些统计数据可用于监控系统、查找性能瓶颈（即性能分析）并预测未来的系统负载（即容量规划）等。

- 优点:

由C语言编写(性能很好 可移植性高)、可运行在嵌入式系统上、包含100多种插件(并提供强大的网络特性)

- 缺点:

本身不能生成图形、监控功能只能进行简单的门阀检测()

- AMP

应用程序性能监控系统(Application Performance Monitoring, APM)

其中Collectd负责收集系统健康度信息(机器内存、CPU使用情况、Disk 读写情况等)，InfluxDB负责基于时序存储收集到的数据，Grafana负责监控数据的图形化展示。

二.使用

- Collectd

- 下载

```
wget https://storage.googleapis.com/collectd-tarballs/collectd-5.7.2.tar.bz2
```

```
tar xf collectd-5.7.2.tar.bz2
```

- 编译

```
cd collectd-5.7.2
```

```
./configure #编译
```

```
--prefix=/usr --sysconfdir=/etc --localstatedir=/var --libdir=/usr/lib --  
mandir=/usr/share/man --enable-all-plugins(可在编译的同时加一些选项)
```

- 安装

```
make all install #安装 (或者直接执行命令安装 sudo apt-get install collectd)
```

目录结构

- 配置文件目录

配置文件目录: `/etc/collectd/collectd.conf`

- 启动文件目录

启动文件目录: `/etc/init.d/collectd`

- 日志文件目录

日志文件目录: `/var/log/syslog`

- 数据存储目录

数据存储目录: `/var/lib/collectd/rrd/`

配置

```
sudo vim /opt/collectd/etc/collectd.conf
```

通过其中的`LoadPlugin section`部分控制使能哪些插件（默认开启了rrdtool插件和syslog插件）

其中`#`符号的数量表示不同含义:

两个`##`开头表示该插件还没有构建,也就不会使能。

一个`#`开头表示该插件已经构建,但是不使能。

没有`#`表示该插件已经构建且使能。

监控系统所用的插件有:cpu,memory,processes,load,interface,disk,swap等

一般只需要修改**network**插件这一项

因为网络插件可以发送到collectd的远程实例中(数据库, csv文件或缓存等介质中),或者接受从远程服务端发来的数据

配置collectd将数据发往InfluxDB

```
Hostname    "127.0.0.1"
FQDNLookup  true
BaseDir     "/var/lib/collectd"
PIDFile     "/var/run/collectd.pid"
PluginDir   "/usr/lib64/collectd"
TypesDB     "/usr/share/collectd/types.db"
LoadPlugin  syslog
LoadPlugin  cpu
LoadPlugin  disk
LoadPlugin  interface
LoadPlugin  memory
LoadPlugin  rrdtool
LoadPlugin  swap
LoadPlugin  network      #去掉#就表示载入该插件
```

```

<Plugin network>
#      # client setup:
      Server "127.0.0.1" "25826" #该地址和端口是接收数据的服务器的地址和端口
      #例如:安装数据库(influxdb等)的服务器的地址和端口,在此处用本地的机器作为数据库
Server

#      <Server "239.192.74.66" "25826"> #若需传输加密数据,就配置这一小块部分
#          SecurityLevel Encrypt
#          Username "user"
#          Password "secret"
#          Interface "eth0"
#          ResolveInterval 14400
#      </Server>
#      TimeToLive 128
#
#      # server setup:
#      Listen "ff18::efc0:4a42" "25826" #如配置这一部分,表示接收从其他collectd实例中
发来的数据
#      <Listen "239.192.74.66" "25826">
#          SecurityLevel sign
#          AuthFile "/etc/collectd/passwd"
#          Interface "eth0"
#      </Listen>
#      MaxPacketSize 1452
#
#      # proxy setup (client and server as above):
#      Forward true
#
#      # statistics about the network plugin itself
#      ReportStats false
#
#      # "garbage collection"
#      CacheFlush 1800
</Plugin>

<Plugin cpu>
    ReportByCpu true
    ReportByState true
    ValuesPercentage true
</Plugin>

<Plugin interface>
    Interface "eth0"
    IgnoreSelected false
</Plugin>

<Plugin load>
    ReportRelative true
</Plugin>

<Plugin network>
    Server "127.0.0.1" "25826"
</Plugin>

<Plugin rrdtool>
    DataDir "/var/lib/collectd/rrd"
</Plugin>

```

ps:若要监控cluster(集群),则每台服务器都需要下载安装collectd并进行相应的配置。

启动

执行命令 `sudo /etc/init.d/collectd start` 或者 `systemctl start collectd`

并设置开机启动 `systemctl enable collectd`

查看数据

若开启了rrdtool插件就可在 `/var/lib/collectd/rrd/` 目录下看到相应的统计数据。

通过下面命令可以查看具体数据：

```
rrdtool fetch *.rrd AVERAGE
```

*.rrd表示任何以.rrd结尾的文件，rrdtool命令更详细的用法可以自行百度。

能够看到第一列为timestamp(时间戳)

• InfluxDB

InfluxDB简介

InfluxDB是用Go语言编写的高性能、高可用的分布式时序数据存储数据库，无其他依赖，安装简单快速。

• 下载

```
wget https://dl.influxdata.com/influxdb/releases/influxdb\_1.0.2\_amd64.deb
```

```
sudo dpkg -i influxdb_1.0.2_amd64.deb
```

• 配置文件目录

```
/etc/influxdb/influxdb.conf
```

Note: 启动后TCP端口:**8083** 为InfluxDB 管理控制台

TCP端口:**8086** 为客户端和InfluxDB通信时的**HTTP** API 启动后InfluxDB用户认证默认是关闭

• 启动InfluxDB

```
systemctl start influxdb.service #启动
```

```
systemctl enable influxdb.service #配置influxdb开机自启
```

• 配置数据库

先创建创建用户:sherlock sherlock 命令行输入influx

```
[root@VM_0_13_centos ~] influx #进入InfluxDB
```

Visit <https://enterprise.influxdata.com> to register for updates, InfluxDB server management, and monitoring.

Connected to <http://localhost:8086> version 0.12.2

InfluxDB shell 0.12.2

```
create database collectdb #创建数据库
```

```
show databases #查看数据库
```

```
name: databases
```

```
-----
```

```
name
```

```
_internal
```

```
collectdb
```

```
> create user sherlock with password 'sherlock' #创建一个用户和密码
```

```
> show users #查看所有用户
```

```
user      admin
```

```
sherlock  false
```

```
> grant all on collectdb to sherlock #把上面创建的数据库的所有权限赋给sherlock用户
```

```
> quit #退出
```

• 启用认证

```
sed -i 's#auth-enabled = false#auth-enabled = true#g' /etc/influxdb/influxdb.conf
```

NOTE!!!

若在此处设置后无法进入InfluxDB数据库 那么需要将/etc/influxdb/influxdb.conf 里面

[http]

```
auth-enabled = false #设置为false
```

否则会发生403错误

重新启动influxdb 服务

```
systemctl restart influxdb.service #重启influxdb
```

• 配置InfluxDB支持Collectd

```
vim /etc/influxdb/influxdb.conf
[[collectd]] # 这个地方是[[[]]]双层的
enabled = true
bind-address = "127.0.0.1:25826"
database = "collectdb"
typesdb = "/usr/share/collectd/types.db" #查找一下types.db文件不一定在这个路径，如果路
径配置错误就不能监听成功
batch-size = 5000
batch-pending = 10
batch-timeout = "10s"
read-buffer = 0
systemctl restart influxdb.service #重启influxdb
```

查看25826 这个端口是否已经监听 如果有 则代表启动正常(可能需要安装netstat 直接安装即可)

```
sudo netstat -anp | grep 25826
```

```
udp    0    0      127.0.0.1:25826      0.0.0.0:*           *** /influxd
```

• 确认数据

进入数据库 influx

Visit <https://enterprise.influxdata.com> to register for updates, InfluxDB server management, and monitoring.

Connected to <http://localhost:8086> version 0.12.2

InfluxDB shell 0.12.2

```
> use collectdb      #使用这个数据库
```

```
Using database collectdb
```

```
> show field keys    #显示所有的keys
```

```
name: cpu_value
```

```
fieldKey
```

```
value
```

```
name: df_free
```

```
fieldKey
```

```
value
```

```
name: df_used
```

```
fieldKey
```

```
value
```

name: disk_read

fieldKey

value

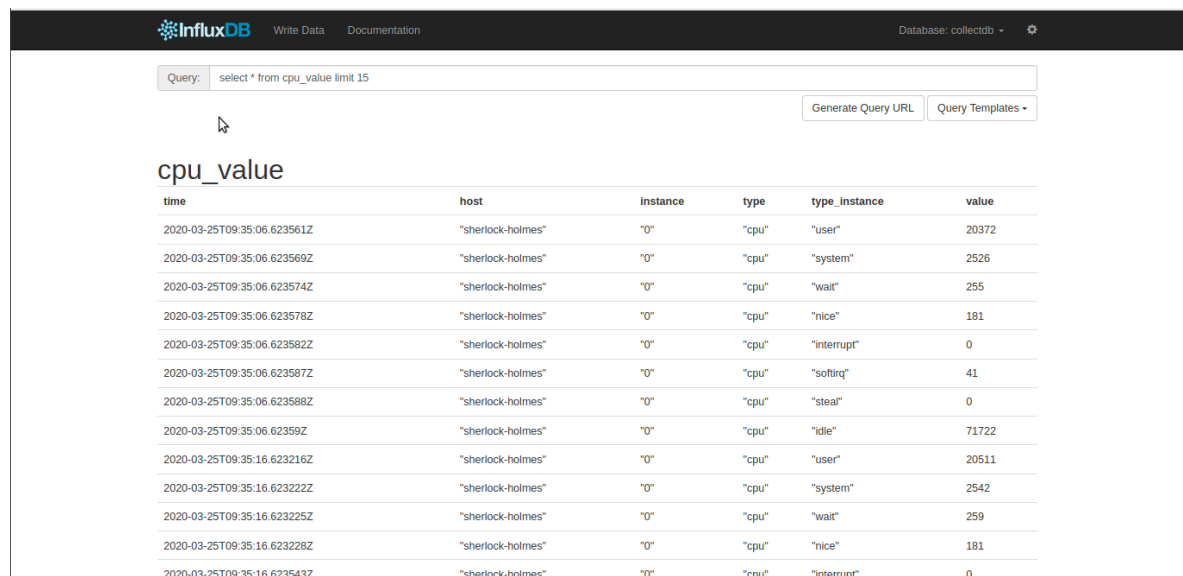
> select * from cpu_value limit 15; #显示15条cpu_value的信息

name: cpu_value

time	host	instance	type	type_instance	value
1461657293000000000	host.example.com	1	cpu	idle	1.59845e+06
1461657293000000000	host.example.com	1	cpu	system	2316
1461657293000000000	host.example.com	1	cpu	nice	508
1461657293000000000	host.example.com	0	cpu	steal	0
1461657293000000000	host.example.com	1	cpu	user	11619
1461657293000000000	host.example.com	1	cpu	interrupt	0
1461657293000000000	host.example.com	1	cpu	steal	0
1461657293000000000	host.example.com	1	cpu	wait	172
1461657293000000000	host.example.com	1	cpu	softirq	0
1461657303000000000	host.example.com	1	cpu	wait	172
1461657303000000000	host.example.com	1	cpu	softirq	0
1461657303000000000	host.example.com	1	cpu	nice	508
1461657303000000000	host.example.com	0	cpu	idle	1.587007e+06
1461657303000000000	host.example.com	0	cpu	softirq	127
1461657303000000000	host.example.com	0	cpu	interrupt	54

在地址栏输入<http://127.0.0.1:8083> 设置账户后 在Query查看数据

select * from cpu_value limit 15 # 显示前15条数据



The screenshot shows the InfluxDB web interface. At the top, there's a navigation bar with 'InfluxDB', 'Write Data', and 'Documentation' links. On the right, it says 'Database: collectdb'. Below the navigation bar, there's a query input field containing 'select * from cpu_value limit 15'. To the right of the input field are buttons for 'Generate Query URL' and 'Query Templates'. Below the query input, the title 'cpu_value' is displayed. Underneath the title is a table with 7 columns: 'time', 'host', 'instance', 'type', 'type_instance', and 'value'. The table contains 15 rows of data, showing various CPU metrics like 'user', 'system', 'wait', 'nice', 'interrupt', 'steal', and 'softirq' for the host 'sherlock-holmes'.

time	host	instance	type	type_instance	value
2020-03-25T09:35:06.623561Z	"sherlock-holmes"	"0"	"cpu"	"user"	20372
2020-03-25T09:35:06.623569Z	"sherlock-holmes"	"0"	"cpu"	"system"	2526
2020-03-25T09:35:06.623574Z	"sherlock-holmes"	"0"	"cpu"	"wait"	255
2020-03-25T09:35:06.623578Z	"sherlock-holmes"	"0"	"cpu"	"nice"	181
2020-03-25T09:35:06.623582Z	"sherlock-holmes"	"0"	"cpu"	"interrupt"	0
2020-03-25T09:35:06.623587Z	"sherlock-holmes"	"0"	"cpu"	"softirq"	41
2020-03-25T09:35:06.623588Z	"sherlock-holmes"	"0"	"cpu"	"steal"	0
2020-03-25T09:35:06.62359Z	"sherlock-holmes"	"0"	"cpu"	"idle"	71722
2020-03-25T09:35:16.623216Z	"sherlock-holmes"	"0"	"cpu"	"user"	20511
2020-03-25T09:35:16.623222Z	"sherlock-holmes"	"0"	"cpu"	"system"	2542
2020-03-25T09:35:16.623225Z	"sherlock-holmes"	"0"	"cpu"	"wait"	259
2020-03-25T09:35:16.623228Z	"sherlock-holmes"	"0"	"cpu"	"nice"	181
2020-03-25T09:35:16.623543Z	"sherlock-holmes"	"0"	"cpu"	"interrupt"	0

• Grafana

• 安装

sudo yum install https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana-5.1.3-1.x86_64.rpm

• 目录结构

/usr/sbin/grafana-server
/etc/init.d/grafana-server 上述命令的拷贝，启动脚本
/etc/sysconfig/grafana-server 环境变量
/etc/grafana/grafana.ini 配置文件
/var/log/grafana/grafana.log 日志文件
/var/lib/grafana/grafana.db sqlite3数据库

• 启动

systemctl start grafana-server.service #启动

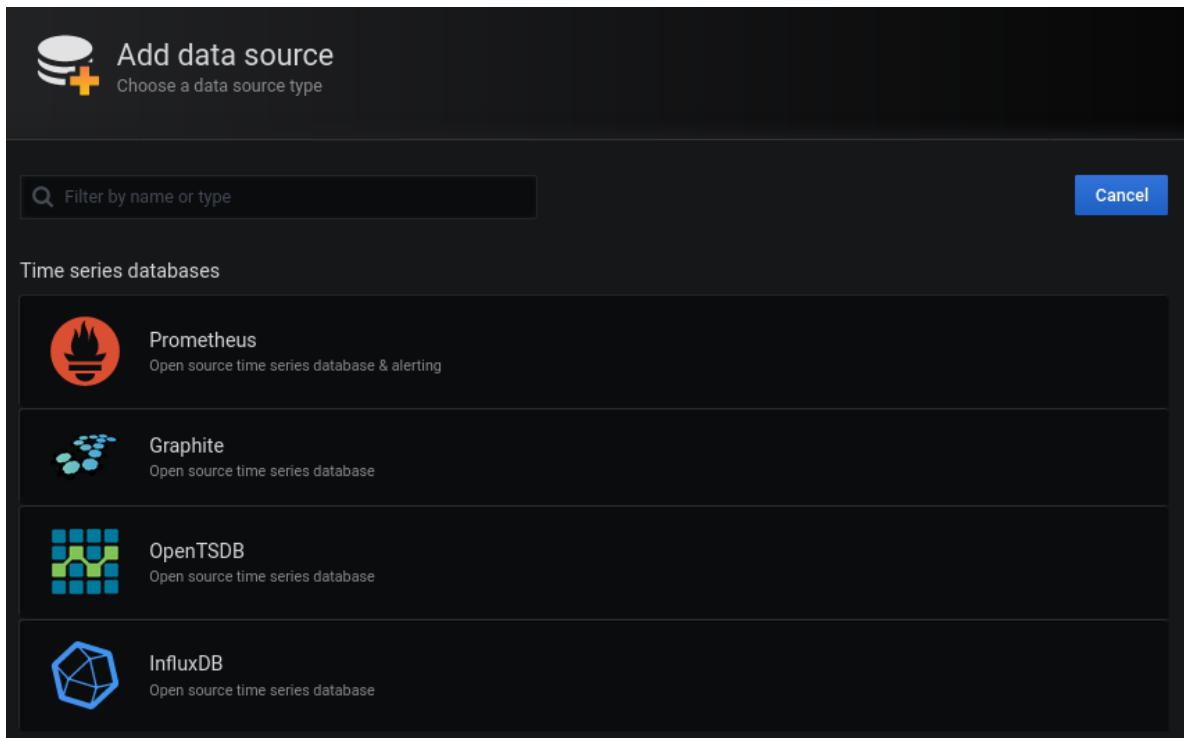
systemctl enable grafana-server.service #配置开机自启

• 配置

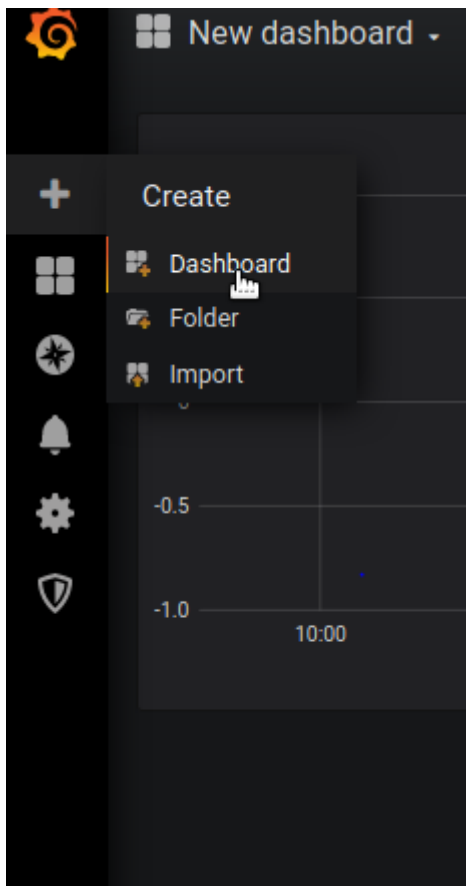
<http://127.0.0.1:3000>默认账号为 `admin` 密码 `admin`

Note:第一次进去会让重新设置账号和密码

Create a data source 选择InfluxDB

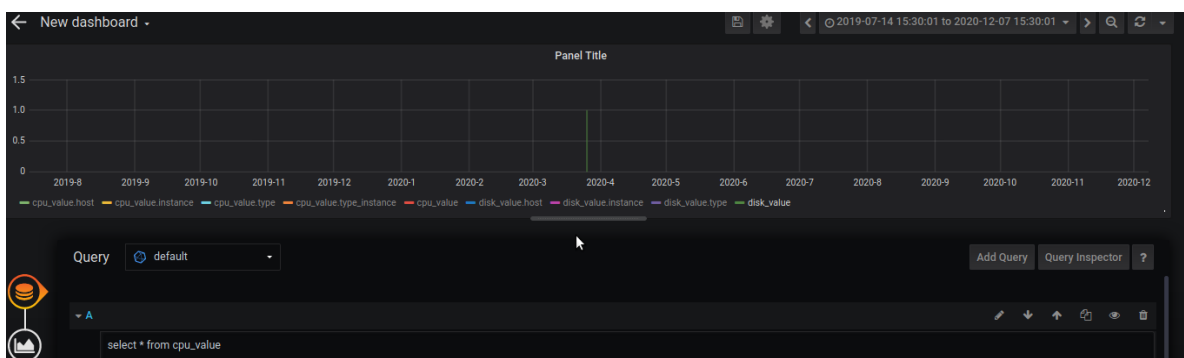
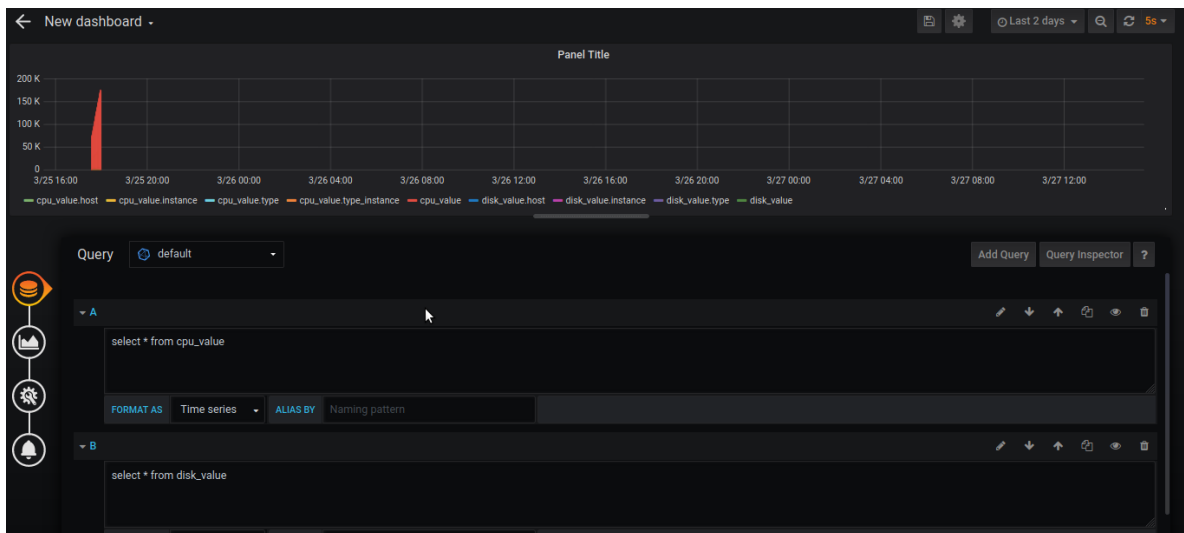


添加InfluxDB数据源



select * from cpu_value (显示CPU的信息)

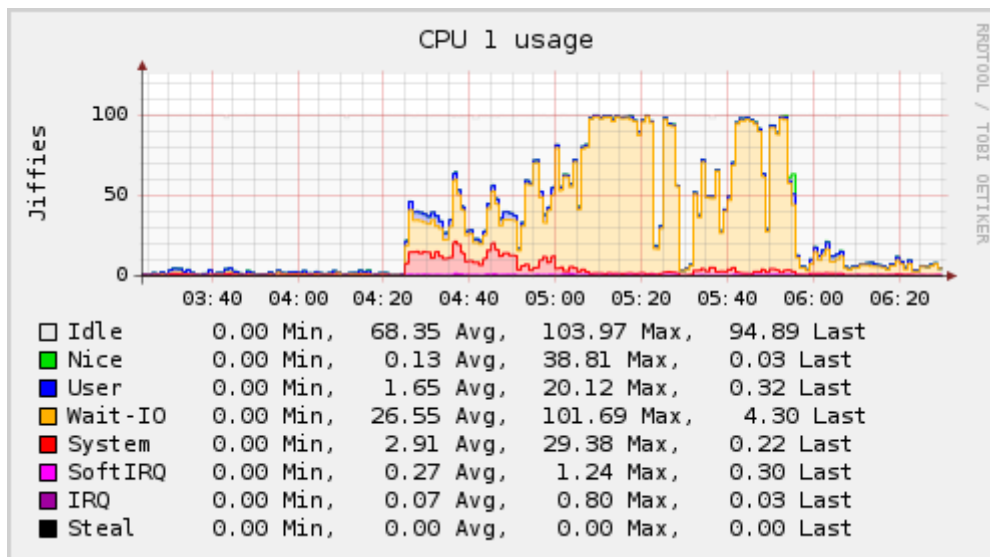
select * from disk_value (显示Disk的信息)



三.主要插件介绍

- Plugin: CPU

可视化显示(使用PHP_GD插件)



cpu

指标释义

jiffies: 是一个单位, jiffies是内核中的一个全局变量, 用来记录自系统启动以来产生的节拍数, 在linux中, 一个节拍大致可理解为操作系统进程调度的最小时间单位, 不同linux内核可能值有不同, 通常在1ms到10ms之间

user: 从系统启动开始累计到当前时刻, 处于用户态的运行时间, 不包含 nice值为负的进程。

nice: 从系统启动开始累计到当前时刻, nice值为负的进程所占用的CPU时间。

idle: 从系统启动开始累计到当前时刻, 除I/O等待时间以外的其它等待时间。

wait-io: 从系统启动开始累计到当前时刻, I/O操作等待时间。

system: 从系统启动开始累计到当前时刻, 处于内核态的运行时间。

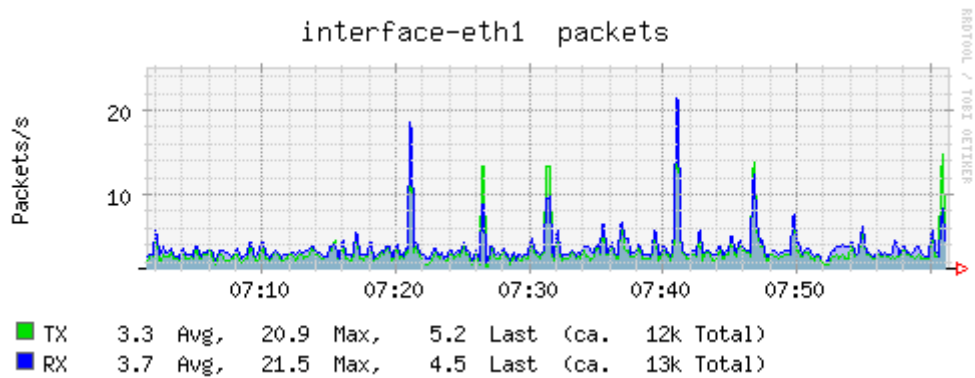
softIRQ: 从系统启动开始累计到当前时刻, 软中断时间。

IRQ:从系统启动开始累计到当前时刻, 硬中断时间。

steal:运行在虚拟环境中其他操作系统所花费的时间。

- Plugin: interface

Interface插件收集关于流量（每秒八位字节），每秒的数据包和接口错误（一秒钟内）的信息。



interface

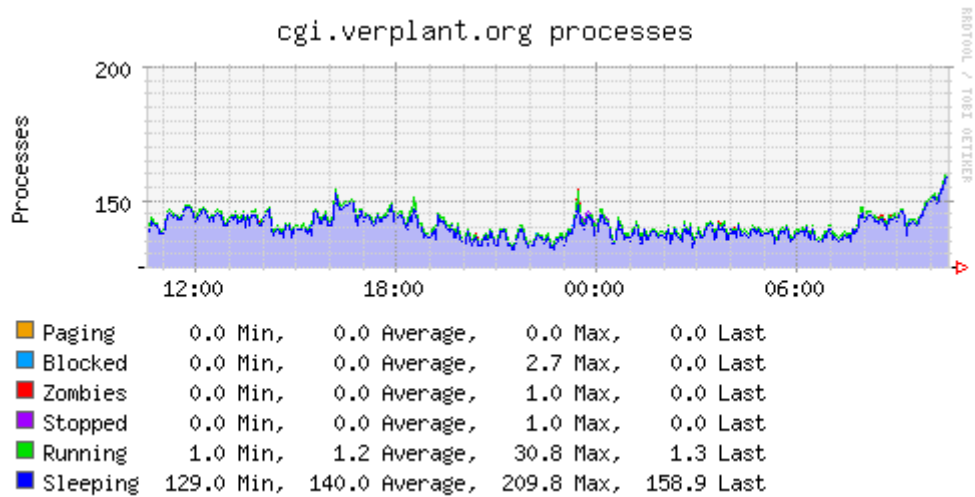
rxpck/s: 每秒钟接收的数据包

txpck/s: 每秒钟发送的数据包

rxbyt/s: 每秒钟接收的字节数

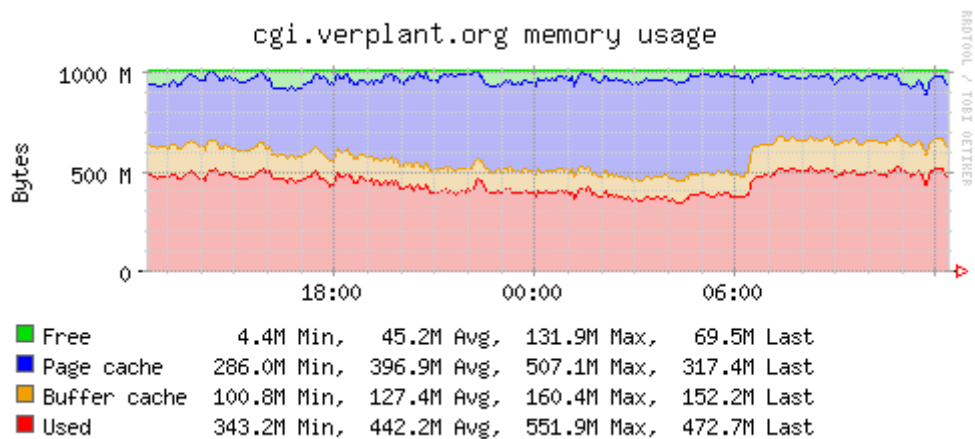
txbyt/s: 每秒钟发送的字节数

- Plugin: processes



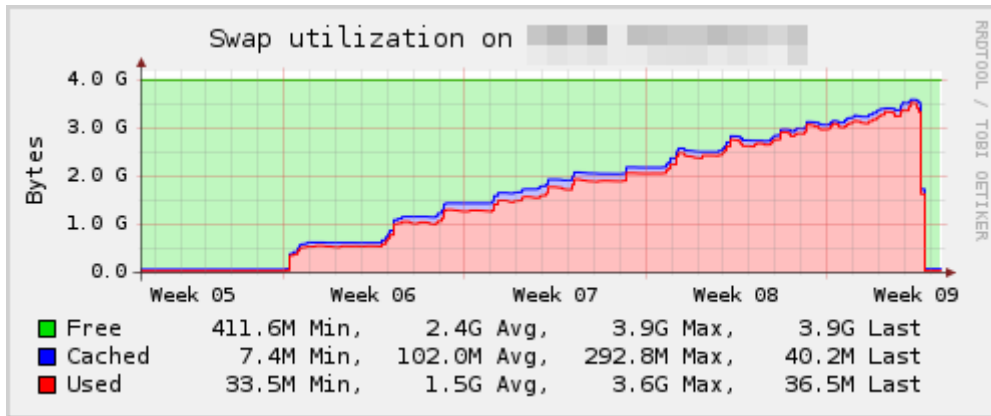
processes

- Plugin: memory 收集系统的物理内存利用率



processes

- Plugin: swap 收集**swap**空间的使用情况



swap

- Plugin: df 统计文件系统的使用信息
- Plugin: irq 收集操作系统处理中断的数量
- Plugin: disk 收集磁盘的性能统计信息
- Plugin: load 收集系统负载信息，定义为队列中可运行任务的数量

在Linux中，系统相关数据可以通过命令 `cd /proc/` 在该目录下查看。

可视化显示

安装**Apache Web Server(127.0.0.1:80)**

1.sudo apt update 2.sudo apt install apache

2.检验是否安装成功 `apache2 -version`

3.配置UFW防火墙 允许外部访问我们系统的某些Web端口 1)sudo ufw app list

Available applications: Apache Apache Full Apache Secure CUPS 2)

`sudo ufw allow 'Apache'` Status: inactive

4.在系统上运行 `sudo systemctl status apache2`

在浏览器中输入 127.0.0.1:80 5.Apache2

查看状态: `service apache2 status/start/stop/restart`

目录:

`/var/www` 安装目录:

`/etc/apache2/` 全局配置:

`/etc/apache2/apache2.conf` 监听端口:

`/etc/apache2/ports.conf` 虚拟主机:

`/etc/apache2/sites-enabled/000-default.conf`

修改权限 `sudo chmod 777 /var/www`

安装**php**

1.sudo apt-get install php

2.其他模块 `sudo apt-get install libapache2-mod-php`

3.重启apache2服务 `sudo systemctl restart apache2`

安装php-gd(用于画图) `sudo apt-get install php7.2-gd` 然后重启

Collectd前端程序

1.参照官网Collectd前端列表 http://collectd.org/wiki/index.php/List_of_front-ends 选择第一个Collectd Graph Panel

wget <http://pommi.nethuis.nl/storage/software/cgp/cgp-0.3.tgz>

`tar -zxvf cgp-0.3.tgz`

`mv cgp-0.3 /var/www/html/cgp` (放到Apache2目录下，默认apache document root在/var/www/html)

2.修改配置文件 `cd /var/www/html/cgp/conf vim config.php`

`$CONFIG['datadir'] = '/var/lib/collectd/rrd';` # 因为启用了network插件，这个位置会变成这样

3.在浏览器输入 <http://127.0.0.1/cgp> 即可

8.自定义插件

collectd插件系统接口

函数	描述
<code>collectd.register_config()</code>	方法用于实现对模块参数的读取
<code>collectd.register_read()</code>	方法用于进行数据的采集
<code>collectd.register_write</code>	方法用于对草技术据进行写处理

Collectd 内部与有自己的类型 存储在Types.db中 ,内容如下:

```
absolute          value:ABSOLUTE:0:U
apache_bytes      value:DERIVE:0:U
apache_connections value:GAUGE:0:65535
apache_idle_workers value:GAUGE:0:65535
apache_requests   value:DERIVE:0:U
apache_scoreboard value:GAUGE:0:65535
ath_nodes         value:GAUGE:0:65535
ath_stat          value:DERIVE:0:U
backends          value:GAUGE:0:65535
bitrate           value:GAUGE:0:4294967295
blocked_clients   value:GAUGE:0:U
bytes             value:GAUGE:0:U
cache_eviction     value:DERIVE:0:U
cache_operation    value:DERIVE:0:U
cache_ratio       value:GAUGE:0:100
cache_result       value:DERIVE:0:U
cache_size        value:GAUGE:0:U
```

其中第一列为类型的名称;

第二列为对类型的描述(包括: GAUGE, DERIVE, COUNTER, ABSOLUTE),

由四列组成, 分别为数据源名、类型、最小值和最大值组成, 中间以冒号“:”进行分隔。

U表示没有定义, 意思就是没有范围。用户可以根据需要定义自己的collectd的类型

综上所述我们可以自己定义一个数据库(在如图所示的路径下)

```
#TypesDB "/usr/share/collectd/types.db"
TypesDB "/usr/share/collectd/mytypes.db"
```

加入如下信息: test value:GAUGE:0:U

然后修改collectd.conf启用自定义的mytypes.db文件

这样就成功添加了一个自定义的类型 test。

利用python语言开发自己插件

- 在配置文件中启用插件

```
LoadPlugin processes
#LoadPlugin protocols
LoadPlugin python
#LoadPlugin redis
#LoadPlugin rrdcached
```

```

<Plugin python>
  ModulePath "/root"
#   LogTraces true
#   Interactive true
  Import "test"
#
  <Module test>
    t "12345"
    Interval 30
  </Module>
</Plugin>

```

配置插件的路径为"/root"，并通过Import将写好的test.py模块进行加载。

如果有多个模块可以调用多个Import。然后配置模块,模块名和Import相对应。

在模块中可以根据插件的需要配置自己的参数，在此处为了随便配置了参数t和Interval。

- 开发自己的python程序

创建test.py文件，并且放到"/root"路径下。

在test.py文件中引入collectd模块。

collectd提供了基本的方法用于注册自己插件的配置加载函数、数据读取函数和数据写函数。

配置文件回调函数

```

def configure_callback(conf=None): # 创建读取配置的函数
    pass
os.system("echo '"+str(conf)+"'>>/tmp/config.log")
try:
    for c in conf.children:
        os.system("echo '"+str(c.values[0])+"'>>/tmp/config.log")
    except Exception ,e:
        os.system("echo '"+e+"'>>/tmp/config.log")
finally:
    pass

```

configure_callback作用是读取collectd.conf文件中的配置，对应的就是上面配置的module,t,Interval。

为了能够看到效果，我们将获取到的配置信息写入到config.log文件中。

读回调函数


```
def getTestValue():
    return 123 # 在真实的系统中，需要根据自己去获取数据

def read_callback(): # 创建的监控的读回调函数
    type = "test" # 设置了类型为自定义的类型“test” 也可以为在types.db中定义的其它类型。
    val = collectd.values(type)
    val.plugin = "test"
    val.host = "127.0.0.1"
    val.type_instance = "test_instance"
    val.interval = 30
    val.values = [getTestValue()]
    val.dispatch() # 最后调用dispatch方法，发送获取的结果 给write插件读取
```

写回调函数

```
def write_callback(data=None):
    os.system("echo '"+str(data)+"'>>/tmp/write.log") # 将数据写 到write.log 文件中
```

在实际的项目中，可以在此处加入自己的逻辑，将数据写入数据库，缓存等

注册回调函数

```
collectd.register_config(configure_callback)
collectd.register_read(read_callback)
collectd.register_write(write_callback)
```

将自己实现的回调函数注册到collectd中，使其生效。

重启服务 # `sudo systemctl restart collectd.service`

将会在/tmp路径下看到config.log 和 write.log。

其中config.log中将会读取 t 和 Interval参数，而在write.log中将会看到test类型 值为123的数据。

END