## 案例八：傻瓜运维工具

做运维的这些年，多多少少积累了一些自己的客户（运维兼职），虽然目前我已经专职做了Linux运维培训，但是平时的运维工作还是不少的，这些客户里服务器量多的十几台，少的就一台。其实，无论多与少，有很多事情都是周而复始，而且所做的事情就那么几件，摆着手指头都能数过来，除非有特殊需求需要特别对待。正是因为这个特点，所以萌生了写一个傻瓜运维脚本的想法，具体要求如下：

1）当执行脚本时，首先会打印系统的几个状态值：系统时间、负载、CPU使用情况、内存使用情况、磁盘使用情况、网卡流量（5s内均值，需列出所有网卡）

2）然后会列出一个命令列表，用户只需要打对应数字，就可以运行列表中的命令

3）列表中的功能有

a）查看网站访问日志最后100行（假设该服务器上只有一个网站，访问日志路径/data/logs/www.log b）查看mysql的慢查询日志，最后50行（mysql慢查询日志路径/data/mysql/slow.log）

c）查看php-fpm的慢执行日志，最后50行（日志路径/usr/local/php/logs/slow.log）

d）重启php-fpm服务

e）重启Nginx服务

f）查看MySQL队列

g）退出脚本

4）脚本为常驻脚本，按ctrl c组合键或者q/Q退出脚本

知识点一：case逻辑判断

在前面的几个案例中我们使用最频繁要数if逻辑判断，如果当判断的条件非常多，则建议使用case，下面是一个case使用场景示例：

case $name in

zhangsan)

echo "Hello, zhangsan."

;;

lisi)

echo "Hello,lisi."

;;

wangwu)

echo "Hello, wangwu."

;;

zhaoliu)

echo "Hello, zhaoliu."

;;

\*)

echo "Sorry, there is no such person."

;;

esac

说明：case开头，esac结尾，$name可选的值有zhangsan, lisi, zhaoliu, wangwu, 这里的\*指的是除了以上可选值外的其它情况。

知识点二：select用法

本案例中的菜单脚本使用select实现起来非常方便，我们先来看一个简单的例子：

select name in zhangsan lisi wangwu zhaoliu

do

case $name in

zhangsan)

echo "Hello, zhangsan."

;;

lisi)

echo "Hello,lisi."

;;

wangwu)

echo "Hello, wangwu."

;;

zhaoliu)

echo "Hello, zhaoliu."

;;

\*)

echo "Sorry, there is no such person."

;;

esac

done

在select脚本中，要结合case一起使用，执行select脚本后可以自动按照循环条件打印列表，而且只要不特意结束脚本，则脚本无限循环。执行脚本，结果如下：

1) zhangsan

2) lisi

3) wangwu

4) zhaoliu

#? 2

Hello,lisi.

#? 3

Hello, wangwu.

#? 4

Hello, zhaoliu.

#?

1) zhangsan

2) lisi

3) wangwu

4) zhaoliu

#?

说明，这里的’#?’可以更改，需要定义PS3，更改以上脚本内容，如下：

PS3="Please select a number: "

select name in zhangsan lisi wangwu zhaoliu

do

case $name in

zhangsan)

echo "Hello, zhangsan."

;;

lisi)

echo "Hello,lisi."

;;

wangwu)

echo "Hello, wangwu."

;;

zhaoliu)

echo "Hello, zhaoliu."

;;

\*)

echo "Sorry, there is no such person."

;;

esac

done

执行脚本后，结果如下：

1) zhangsan

2) lisi

3) wangwu

4) zhaoliu

Please select a number: 3

Hello, wangwu.

Please select a number: 4

Hello, zhaoliu.

知识点三：w命令

w命令查看当前系统的负载

# w

15:23:46 up 3:34, 2 users, load average: 0.03, 0.05, 0.00

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

root pts/0 10.72.137.53 12:28 1:17m 1:32 1:32 -bash

相信所有的linux管理员最常用的命令就是这个'w'命令了，该命令显示的信息还是蛮丰富的。第一行从左面开始显示的信息依次为：时间，系统运行时间，登录用户数，平均负载。第二行开始以及下面所有的行，告诉我们的信息是，当前登录的都有哪些用户，以及他们是从哪里登录的等等。其实，在这些信息当中，最应该关注的应该是第一行中的'load average:'后面的三个数值。

第一个数值表示1分钟内系统的平均负载值；第二个数值表示5分钟内系统的平均负载值；第三个数值表示15分钟系统的平均负载值。这个值的意义是，单位时间段内CPU活动进程数。当然这个值越大就说明你的服务器压力越大。一般情况下这个值只要不超过服务器的cpu数量就没有关系，如果服务器cpu数量为8，那么这个值若小于8，就说明当前服务器没有压力，否则就要关注一下了。如何查看服务器有几个cpu？

# cat /proc/cpuinfo

processor : 0

vendor\_id : AuthenticAMD

cpu family : 16

model : 6

model name : AMD Phenom(tm) II N660 Dual-Core Processor

stepping : 3

cpu MHz : 3000.000

cache size : 1024 KB

fdiv\_bug : no

hlt\_bug : no

f00f\_bug : no

coma\_bug : no

fpu : yes

fpu\_exception : yes

cpuid level : 5

wp : yes

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx mmxext fxsr\_opt rdtscp lm 3dnowext 3dnow constant\_tsc up tsc\_reliable nonstop\_tsc extd\_apicid pni cx16 popcnt lahf\_lm abm sse4a misalignsse 3dnowprefetch

bogomips : 6000.00

clflush size : 64

cache\_alignment : 64

address sizes : 36 bits physical, 48 bits virtual

power management: ts ttp tm stc 100mhzsteps hwpstate

/proc/cpuinfo这个文件记录了cpu的详细信息。如果服务器是2颗4核cpu，在linux看来，它就是8个cpu。查看这个文件时则会显示8段类似的信息，而最后一段信息中processor : 后面跟的是7。所以查看当前系统有几个cpu，我们可以使用这个命令：grep -c 'processor' /proc/cpuinfo。而如何看几颗物理cpu呢，需要查看关键字&quot;physical id&quot;。

知识点四：free命令

free命令查看内存使用情况（CentOS7系统示例）

# free

total used free shared buff/cache available

Mem: 1876632 571084 82992 98124 1222556 982532

Swap: 3881920 0 3881920

第一行是说明，第二行是内存使用情况，第三行是swap使用情况，我们主要关注第二行。

第一列：total内存总大小 ,单位KB

第二列：used内存已用大小

第三列：free内存剩余大小

第四列：shared共享内存大小

第五列：buff/cache 缓冲/缓存

第六列：available为系统真正的物理剩余内存大小，其值为free+buff/cache剩余的部分

公式：total=used+free+buff/cache

指定单位为M时，使用free -m，指定单位为G时，使用free -g或者free -h在后面显示单位

# free -h

total used free shared buff/cache available

Mem: 1.8G 557M 80M 95M 1.2G 959M

Swap: 3.8G 0B 3.8G

知识点五：sar命令

sar命令很强大，它可以监控系统所有资源状态，比如平均负载、网卡流量、磁盘状态、内存使用等等。它不同于其他系统状态监控工具的地方在于，它可以打印历史信息，可以显示当天从零点开始到当前时刻的系统状态信息。如果你系统没有安装这个命令，请使用&quot;yum install -y sysstat&quot;命令安装。初次使用sar命令会报错，那是因为sar工具还没有生成相应的数据库文件（实时监控就不会了，因为不用去查询那个库文件）。它的数据库文件在/var/log/sa/目录下，默认保存一个月。因为这个命令太过复杂，所以只介绍几个。

1）查看网卡流量sar -n DEV

# sar -n DEV 1 2

Linux 3.10.0-693.el7.x86\_64 (linux7-128) 2018年10月03日 \_x86\_64\_ (2 CPU)

22时03分13秒 IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s

22时03分14秒 lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00

22时03分14秒 virbr0-nic 0.00 0.00 0.00 0.00 0.00 0.00 0.00

22时03分14秒 virbr0 0.00 0.00 0.00 0.00 0.00 0.00 0.00

22时03分14秒 ens33 3.00 1.00 0.18 0.19 0.00 0.00 0.00

22时03分14秒 IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s

22时03分15秒 lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00

22时03分15秒 virbr0-nic 0.00 0.00 0.00 0.00 0.00 0.00 0.00

22时03分15秒 virbr0 0.00 0.00 0.00 0.00 0.00 0.00 0.00

22时03分15秒 ens33 3.00 2.00 0.18 0.64 0.00 0.00 0.00

平均时间: IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s

平均时间: lo 0.00 0.00 0.00 0.00 0.00 0.00 0.00

平均时间: virbr0-nic 0.00 0.00 0.00 0.00 0.00 0.00 0.00

平均时间: virbr0 0.00 0.00 0.00 0.00 0.00 0.00 0.00

平均时间: ens33 3.00 1.50 0.18 0.42 0.00 0.00 0.00

后面跟1 2 表示每隔1秒显示一次，显示2次

第一列：时间

第二列：IFACE 网卡名字

第三列：rxpck/s 接受到的数据包，单位个数，几千正常，上万就不正常

第四列：txpck/s 发送出去的数据包

第五列：rxkB/s 接收的数据量，单位KB

另外也可以查看某一天的网卡流量历史，使用-f选项，后面跟文件名，如果你的系统格式Redhat或者CentOS那么sar的库文件一定是在/var/log/sa/目录下的。

# sar -n DEV -f /var/log/sa/sa24

Linux 3.10.0-693.el7.x86\_64 2018年09月24日 \_x86\_64\_(2 CPU)

00时00分01秒 IFACE rxpck/s txpck/s rxkB/s txkB/s rxcmp/s txcmp/s rxmcst/s

00时10分01秒 eth0 59.65 70.60 48.32 7.11 0.00 0.00 0.00

00时10分01秒 eth1 132.86 113.86 22.49 172.84 0.00 0.00 0.00

00时10分01秒 lo 4.09 4.09 0.46 0.46 0.00 0.00 0.00

00时20分01秒 eth0 50.54 59.29 39.08 6.18 0.00 0.00 0.00

00时20分01秒 eth1 92.71 80.66 13.45 123.28 0.00 0.00 0.00

00时20分01秒 lo 3.01 3.01 0.34 0.34 0.00 0.00 0.00

00时30分01秒 eth0 46.54 53.23 37.88 5.91 0.00 0.00 0.00

00时30分01秒 eth1 95.63 84.75 13.95 133.82 0.00 0.00 0.00

00时30分01秒 lo 3.23 3.23 0.40 0.40 0.00 0.00 0.00

00时40分01秒 eth0 33.75 37.70 32.68 4.59 0.00 0.00 0.00

00时40分01秒 eth1 89.37 76.51 11.25 135.34 0.00 0.00 0.00

00时40分01秒 lo 2.09 2.09 0.25 0.25 0.00 0.00 0.00

2）查看历史负载

＃ sar -q

Linux 3.10.0-693.el7.x86\_64 2018年10月03日 \_x86\_64\_(2 CPU)

00时00分01秒 runq-sz plist-sz ldavg-1 ldavg-5 ldavg-15 blocked

00时10分01秒 2 505 1.37 1.33 1.23 2

00时20分01秒 5 505 1.43 1.39 1.29 1

00时30分01秒 1 506 1.17 1.34 1.35 2

00时40分01秒 1 506 1.23 1.51 1.48 2

00时50分01秒 2 500 0.91 1.32 1.43 0

01时00分01秒 1 506 0.43 0.63 1.01 1

01时10分01秒 1 506 1.22 1.41 1.29 1

01时20分01秒 2 506 1.29 1.36 1.33 1

01时30分01秒 1 506 1.80 1.75 1.49 1

01时40分01秒 3 506 1.40 1.54 1.50 1

01时50分01秒 1 502 1.28 1.39 1.43 0

这个命令有助于我们查看服务器在过去的某个时间的负载状况。qunq-sz指的是run状态的进程数量，plist-sz指的是进程数量，ldavg-1指的是1分钟内平均负载值，ldavg-5指的是5分钟内平均负载值，ldavg-15指的是15分钟内平均负载值， blocked指的是被阻塞的进程数。

sar不带任何选项，用来查看cpu使用情况

＃ sar 1 5

15时23分28秒 CPU %user %nice %system %iowait %steal %idle

15时23分29秒 all 2.51 0.00 0.50 0.00 0.00 96.98

15时23分30秒 all 2.49 0.00 1.00 0.00 0.00 96.52

15时23分31秒 all 3.54 0.00 1.01 0.00 0.00 95.45

15时23分32秒 all 2.51 0.00 0.50 0.00 0.00 96.98

15时23分33秒 all 2.50 0.00 0.50 0.00 0.00 97.00

平均时间: all 2.71 0.00 0.70 0.00 0.00 96.59

知识点六：MySQL慢查询日志

MySQL的慢查询日志用来记录在MySQL中响应时间超过阈值的语句，具体指运行时间超过long\_query\_time值的SQL，则会被记录到慢查询日志中。默认情况下，MySQL数据库是不开启慢查询日志的，long\_query\_time的默认值为10（即10秒，建议设置为1秒），即运行10秒以上的语句是慢查询语句。

通过分析MySQL的慢查询日志让我们找到那些查询很慢的SQL语句，然后就可以针对这些慢查询SQL进行优化，通常情况下，造成慢查询的原因往往是因为表没有创建索引。如下为MySQL5.7版本my.cnf配置慢查询日志的参数：

slow\_query\_log = on

long\_query\_time = 1

slow\_query\_log\_file = /data/mysql/slow.log

如下为慢查询日志示例：

# Time: 181003 20:14:39

# User@Host: 552\_com[552\_com] @ web109 [192.168.15.9] Id: 868114397

# Query\_time: 4.814700 Lock\_time: 0.000069 Rows\_sent: 500 Rows\_examined: 3940362

SET timestamp=1538568879;

SELECT tid, fid, subject, dateline, lastpost, replies, views, digest

FROM cdb\_threads

WHERE lastpost > 1538482469 AND displayorder >= 0

LIMIT 500;

知识点七：php-fpm慢执行日志

对于LNMP架构的网站如果访问卡顿，我们第一时间应该想到查看php-fpm的慢执行日志，该日志和MySQL慢查询日志类似，它会记录执行慢的php代码，可以说php-fpm的慢执行日志是追踪网站性能的利器。在php-fpm.conf中加上两行配置即可：

request\_slowlog\_timeout = 1

slowlog = /usr/local/php/logs/slow.log

慢执行日志示例：

script\_filename = /data/wwwroot/www.wen.com/forum.php

[0x00007f9b8c197ea8] curl\_exec() /data/wwwroot/www.wen.com/source/function/function\_filesock.php:68

[0x00007f9b8c197c90] \_dfsockopen() /data/wwwroot/www.wen.com/source/function/function\_core.php:201

[0x00007f9b8c197b30] dfsockopen() /data/wwwroot/www.wen.com/source/class/class\_captcha.php:93

[0x00007f9b8c1976b8] isneed() /data/wwwroot/www.wen.com/source/class/helper/helper\_seccheck.php:300

[0x00007f9b8c1975d0] seccheck() /data/wwwroot/www.wen.com/source/function/function\_core.php:1349

[0x00007f9b8c180a30] seccheck() /data/wwwroot/www.wen.com/source/module/forum/forum\_forumdisplay.php:915

[0x00007f9b8c180130] +++ dump failed

知识点八：shell脚本中显示颜色

为了让shell脚本在执行过程中输出的信息更加容易分辨，我们会特意给它带上颜色。先来看一个小例子：

# echo -e "\033[31m 红色字 \033[0m"

红色字

这样输出的字颜色为红色，常用的几种颜色总结如下：

echo -e "\033[30m 黑色字 \033[0m"

echo -e "\033[31m 红色字 \033[0m"

echo -e "\033[32m 绿色字 \033[0m"

echo -e "\033[33m 黄色字 \033[0m"

echo -e "\033[34m 蓝色字 \033[0m"

echo -e "\033[35m 紫色字 \033[0m"

echo -e "\033[36m 天蓝字 \033[0m"

echo -e "\033[37m 白色字 \033[0m"

本案例参考脚本

#!/bin/bash

#这是一个傻瓜运维脚本，根据列表输入对应数字即可实现想要功能

#作者：

#日期：2018-10-04

#版本：v0.1

LANG=en

sar 1 5 > /tmp/cpu.log &amp;

sar -n DEV 1 5 |grep '^Average:' > /tmp/net.log &amp;

echo -n "收集数据"

for i in `seq 1 5`

do

echo -n "."

sleep 1

done

echo

t=`date +"%F %T"`

load=`uptime |awk -F 'load averages?: ' '{print $2}'|cut -d '.' -f1`

cpu\_idle=`tail -1 /tmp/cpu.log|awk '{print $NF}'`

cpu\_use=`echo "scale=2;100-$cpu\_idle"|bc`

mem\_tot=`free -m |grep '^Mem:'|awk '{print $2}'`

mem\_ava=`free -m |grep '^Mem:'|awk '{print $NF}'`

mysql\_p="dR6wB1jzq"

echo -e "\033[32m当前时间：$t \033[0m"

echo "######"

echo -e "\033[31m当前负载：$load \033[0m"

echo "######"

echo -e "\033[33mCPU使用率：$cpu\_use% \033[0m"

echo "######"

echo -e "\033[34m内存总数：$mem\_tot"MB", 内存剩余：$mem\_ava"MB" \033[0m"

echo "######"

echo -e "\033[35m磁盘空间使用情况：\033[0m"

df -h

echo "######"

echo -e "\033[36m磁盘inode使用情况：\033[0m”

df -i

echo "######"

sed '1d' /tmp/net.log |awk '{print "网卡"$2": 入口流量"$5/1000\*8"Mbi, 出口流量"$6/1000\*8"Mbi"}'

echo "######"

get\_acc\_log()

{

tail -100 /data/logs/www.log

}

get\_mysql\_slow\_log()

{

tail -50 /data/mysql/slow.log

}

get\_php\_slow\_log()

{

tail -50 /usr/local/php/logs/slow.log

}

restart\_php()

{

/etc/init.d/php-fpm restart

}

restart\_nginx()

{

/etc/init.d/nginx restart

}

get\_mysql\_process()

{

mysql -uroot -p$mysql\_p -e "show processlist"

}

PS3="请选择你想要做的操作："

select c in 查看访问日志 查看mysql慢查询日志 查看php-fpm的慢执行日志 重启php-fpm服务 重启nginx服务 查看mysql队列 退出脚本

do

case $c in

查看访问日志)

get\_acc\_log

;;

查看mysql慢查询日志)

get\_mysql\_slow\_log

;;

查看php-fpm的慢执行日志)

get\_php\_slow\_log

;;

重启php-fpm服务)

restart\_php

;;

重启nginx服务)

restart\_nginx

;;

查看mysql队列)

get\_mysql\_process

;;

退出脚本)

exit 0

;;

esac

done