# Linux 查询 OS、CPU、内存、硬盘信息

CPP开发者 2020-12-13 03:50



作者：人生的哲理
https://www.cnblogs.com/renshengdezheli/p/13427865.html

## 一.前言

当我们接手了一台或者几台服务器的时候，首先我们有必要对服务器的基本配置有所认识，这样才可以对症下药，对以后的软件部署，系统运维会有事半功倍的效果。

## 二.关于服务器基本配置

查询服务器的基本配置一般查询操作系统，CPU，内存，硬盘，下面进行逐一讲解。

### 2.1 操作系统基本配置查询

查看操作系统版本

```
#cat /etc/redhat-release这个命令主要是查看红帽发行的操作系统的版本号
[root@node5 ~]# cat /etc/redhat-release
CentOS Linux release 7.4.1708 (Core)
#cat /etc/issue这个命令适用于大多数linux发行版
[root@node5 ~]# cat /etc/issue
\S
Kernel \r on an \m
```

查看操作系统内核版本

```
[root@node5 ~]# uname -r
3.10.0-693.el7.x86_64
```

查看操作系统详细信息

```
[root@node5 ~]# uname -a
Linux node5 3.10.0-693.el7.x86_64 #1 SMP Tue Aug 22 21:09:27 UTC 2017
```

```
x86_64 x86_64 x86_64 GNU/Linux
#从上面这段输出可以看出，该服务器主机名是node5，linux内核版本是3.10.0-693.el7.x86_64，CPU是x86架构

#该命令可以查看更多信息
[root@node5 ~]# more /etc/*release
::::::::::::::
/etc/centos-release
::::::::::::::
CentOS Linux release 7.4.1708 (Core)
::::::::::::::
/etc/os-release
::::::::::::::
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"

::::::::::::::
/etc/redhat-release
::::::::::::::
CentOS Linux release 7.4.1708 (Core)
::::::::::::::
/etc/system-release
::::::::::::::
CentOS Linux release 7.4.1708 (Core)
```

## 2.2 CPU基本配置查询

名词解释

| 名词 | 含义 |
| --- | --- |
| CPU物理个数 | 主板上实际插入的cpu数量 |
| CPU核心数 | 单块CPU上面能处理数据的芯片组的数量，如双核、四核等 （cpu cores） |
| 逻 辑 CPU 数 / 线程数 | 一般情况下，逻辑cpu=物理CPU个数×每颗核数，如果不相等的话，则表示服务器的CPU支持超线程技术 |

查看 CPU 物理个数

```
[root@node5 ~]# grep 'physical id' /proc/cpuinfo | sort -u | wc -l
1
```

查看 CPU 核心数量

```
[root@node5 ~]# grep 'core id' /proc/cpuinfo | sort -u | wc -l
4
```

查看 CPU 线程数

```
#逻辑cpu数：一般情况下，逻辑cpu=物理CPU个数×每颗核数，如果不相等的话，则表示服务器的
CPU支持超线程技术（HT：简单来说，它可使处理#器中的1 颗内核如2 颗内核那样在操作系统中发
挥作用。这样一来，操作系统可使用的执行资源扩大了一倍，大幅提高了系统的整体性能，此时逻#
辑cpu=物理CPU个数×每颗核数x2）
[root@node5 ~]# cat /proc/cpuinfo| grep "processor"|wc -l
4
[root@node5 ~]# grep 'processor' /proc/cpuinfo | sort -u | wc -l
4
```

查看 CPU 型号

```
[root@node5 ~]# cat /proc/cpuinfo | grep name | sort | uniq
model name      : Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
[root@node5 ~]# dmidecode -s processor-version | uniq    #使用uniq进行去重
```

```
Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
```

查看 CPU 的详细信息

```
#CPU有几个核，就会输出几个重复的信息
[root@node5 ~]# cat /proc/cpuinfo
processor   : 0
vendor_id   : GenuineIntel
cpu family   : 6
model    : 142
model name   : Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
stepping   : 10
microcode   : 0x96
cpu MHz    : 2000.921
cache size   : 8192 KB
physical id   : 0
siblings   : 4
core id    : 0
cpu cores   : 4
apicid    : 0
initial apicid   : 0
fpu    : yes
fpu_exception   : yes
cpuid level   : 22
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 ss ht syscall nx pdpe1gb
rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable
nonstop_tsc eagerfpu pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c
rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow vnmi ept vpid
fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpcid mpx rdseed adx smap
clflushopt xsaveopt xsavec arat
bogomips   : 4002.00
clflush size   : 64
cache_alignment   : 64
address sizes   : 43 bits physical, 48 bits virtual
```

```
power management:
```

查看CPU的详细信息

```
[root@node5 ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:   0-3
Thread(s) per core:    1
Core(s) per socket:    4
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 142
Model name:            Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
Stepping:              10
CPU MHz:               2000.921
BogoMIPS:              4002.00
Virtualization:        VT-x
Hypervisor vendor:     VMware
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              8192K
NUMA node0 CPU(s):     0-3
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep
mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx
pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology
tsc_reliable nonstop_tsc eagerfpu pni pclmulqdq vmx ssse3 fma cx16
pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave
avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow vnmi
ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpcid mpx rdseed
adx smap clflushopt xsaveopt xsavec arat
```

**CPU配置总结**

通过以上的查询，我们可以知道该服务器是1路4核的CPU ，CPU型号是Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz，该CPU没有超线程。

## 2.3 内存基本配置查询

名词解释

| 名词 | 含义 |
| --- | --- |
| Mem | 内存的使用情况总览表 |
| Swap | 虚拟内存。即可以把数据存放在硬盘上的数据，当物理内存不足时，拿出部分硬盘空间当SWAP分区（虚拟成内存）使用，从而解决内存容量不足的情况。SWAP意思是交换，顾名思义，当某进程向OS请求内存发现不足时，OS会把内存中暂时不用的数据交换出去，放在SWAP分区中，这个过程称为SWAP OUT。当某进程又需要这些数据且OS发现还有空闲物理内存时，又会把SWAP分区中的数据交换回物理内存中，这个过程称为SWAP IN。当然，swap大小是有上限的，一旦swap使用完，操作系统会触发OOM-Killer机制，把消耗内存最多的进程kill掉以释放内存。 |
| shared | 共享内存，即和普通用户共享的物理内存值， 主要用于进程间通信 |
| buffers | 用于存放要输出到disk（块设备）的数据的 |
| cached | 存放从disk上读出的数据 |
| total | 总的物理内存，total=used+free |

| 名词 | 含义 |
| --- | --- |
| used | 使用掉的内存 |
| free | 空闲的内存 |

查询服务器内存

```
[root@node5 ~]# free -m
              total        used        free      shared  buff/cache
available
Mem:           3941         286        3446          19         208
3407
Swap:          2047           0        2047

#注释
#linux的内存管理机制的思想包括（不敢说就是）内存利用率最大化。内核会把剩余的内存申请为cached，而cached不属于free范畴。当系统运#行时间较久，会发现cached很大，对于有频繁文件读写操作的系统，这种现象会更加明显。直观的看，此时free的内存会非常小，但并不代表可##用的内存小，当一个程序需要申请较大的内存时，如果free的内存不够，内核会把部分cached的内存回收，回收的内存再分配给应用程序。所以#对于linux系统，可用于分配的内存不只是free的内存，还包括cached的内存（其实还包括buffers）。
#对于操作系统：
#MemFree=total-used
#MemUsed  = MemTotal - MemFree
#对于应用程序：
#MemFree=buffers+cached+free
```

每隔3秒查询一下内存

```
[root@node5 ~]# free -s 3
              total        used        free      shared  buff/cache
available
Mem:        4036316      361144     3458272       19536      216900
3419776
```

```
Swap:          2097148           0     2097148

               total        used        free      shared  buff/cache
available
Mem:           4036316      361144     3458272       19536      216900
3419776
Swap:          2097148           0     2097148

               total        used        free      shared  buff/cache
available
Mem:           4036316      361144     3458272       19536      216900
3419776
Swap:          2097148           0     2097148
```

## 2.4 硬盘基本配置查询

查询磁盘整体使用情况

```
[root@node5 ~]# df -h
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/centos-root   17G  4.1G   13G  24% /
devtmpfs                 2.0G     0  2.0G   0% /dev
tmpfs                    2.0G  8.0K  2.0G   1% /dev/shm
tmpfs                    2.0G  8.7M  2.0G   1% /run
tmpfs                    2.0G     0  2.0G   0% /sys/fs/cgroup
/dev/sda1               1014M  125M  890M  13% /boot
tmpfs                    395M     0  395M   0% /run/user/0
#命令拓展
#df -a 显示全部的文件系统的使用情况
#df -i显示inode信息
#df -k 已字节数显示区块占用情况
#df -T 显示文件系统的类型
```

查询某个目录磁盘占用情况

```
#命令拓展
#du -s 指定目录大小汇总
```

```
#du -h带计量单位
#du -a 含文件
#du --max-depth=1 子目录深度
#du -c 列出明细的同时，增加汇总值
[root@node5 ~]# du -sh /home/
1.7G  /home/

[root@node5 ~]# du -ach --max-depth=2 /home/
4.0K  /home/www/.bash_logout
4.0K  /home/www/.bash_profile
4.0K  /home/www/.bashrc
4.0K  /home/www/web
16K  /home/www
4.0K  /home/nginx/.bash_logout
4.0K  /home/nginx/.bash_profile
4.0K  /home/nginx/.bashrc
12K  /home/nginx
4.0K  /home/esnode/.bash_logout
4.0K  /home/esnode/.bash_profile
4.0K  /home/esnode/.bashrc
4.0K  /home/esnode/.oracle_jre_usage
4.3M  /home/esnode/elasticsearch-analysis-ik-6.2.2.zip
80M  /home/esnode/kibana-6.2.2-linux-x86_64.tar.gz
300M  /home/esnode/x-pack-6.2.2.zip
28M  /home/esnode/elasticsearch-6.2.2.tar.gz
4.0K  /home/esnode/.bash_history
294M  /home/esnode/elasticsearch-6.2.2
4.0K  /home/esnode/.ssh
4.0K  /home/esnode/x-pack生成的秘钥.txt
1014M  /home/esnode/kibana-6.2.2-linux-x86_64
8.0K  /home/esnode/.viminfo
1.7G  /home/esnode
1.7G  /home/
1.7G  total
```

查看目录结构

```
#tree命令默认没有安装，需要手动安装一下
[root@node5 ~]# yum -y install tree
#-L指定目录深度
[root@node5 ~]# tree -L 2 /home/
/home/
├── esnode
│   ├── elasticsearch-6.2.2
│   ├── elasticsearch-6.2.2.tar.gz
│   ├── elasticsearch-analysis-ik-6.2.2.zip
│   ├── kibana-6.2.2-linux-x86_64
│   ├── kibana-6.2.2-linux-x86_64.tar.gz
│   ├── x-pack-6.2.2.zip
│   └── x-
pack\347\224\237\346\210\220\347\232\204\347\247\230\351\222\245.txt
├── nginx
└── www
    └── web

6 directories, 5 files
```

以树状的格式显示所有可用的块设备信息

```
[root@node5 ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0   20G  0 disk
├─sda1            8:1    0    1G  0 part /boot
└─sda2            8:2    0   19G  0 part
  ├─centos-root 253:0    0   17G  0 lvm  /
  └─centos-swap 253:1    0    2G  0 lvm  [SWAP]
sdb               8:16   0    1G  0 disk
└─sdb1            8:17   0  200M  0 part
sr0              11:0    1 1024M  0 rom

#注释
#NAME -- 设备的名称
#MAJ:MIN -- Linux 操作系统中的每个设备都以一个文件表示，对块（磁盘）设备来说，这里用主次设备编号来描述设备。
```

```
#RM -- 可移动设备。如果这是一个可移动设备将显示 1，否则显示 0。
#TYPE -- 设备的类型
#MOUNTPOINT -- 设备挂载的位置
#RO -- 对于只读文件系统，这里会显示 1，否则显示 0。
#SIZE -- 设备的容量
```

列出所有可用的设备、通用唯一识别码（UUID）、文件系统类型以及卷标

```
[root@node5 ~]# blkid
/dev/sda1: UUID="6503b4ad-2975-4152-a824-feb7bea1b622" TYPE="xfs"
/dev/sda2: UUID="nqZ4uJ-ksnN-KzYS-N42b-00m3-Ohc2-BJXunP"
TYPE="LVM2_member"
/dev/sdb1: UUID="94396e17-4821-4957-aa76-d41f33958ff5" TYPE="xfs"
/dev/mapper/centos-root: UUID="c1d38b37-821d-48e7-8727-3937ccc657a4"
TYPE="xfs"
/dev/mapper/centos-swap: UUID="c2fcaf11-42d8-4e4c-bf9e-6464f0777198"
TYPE="swap"
```

- EOF -

推荐阅读 — 点击标题可跳转

1、C++模板之SFINAE和enable_if分析

2、据说程序员等电梯的时候都想过调度算法，网友：还真是

3、C++ type traits分析