# 内存问题：

## 内存去向分析

**我们查看内存信息的方法如下：**

1. free（后面可以接一些参数，比如-h，-m等）

2）top（可以看到基本的内存信息以及各个进程占用的内存百分比（%Mem）按M排序）

3）ps（可以看到各个进程占用的内存(RSS列)），命令ps -aux --sort=-rss |head -n 20排序

4）cat /proc/meminfo(上面几个命令对于内存的基本信息都是来自于/proc/ meminfo



我们的内存消耗主要在于这几部分：

**Buffers+Cached**：可以回收部分）

**RSS**：各个进程占用的内存

**Slab+PageTables**：内存管理自己的消耗

**Shmem**：free结果中的shared以及/proc/ meminfo中的Shmem，这是tmpfs的一部分，从环境使用情况看，主要是journal日志占用的比较多，存放位置为：/run/log/journal/

**内存模块管理不到的部分**：比如巨页HugePages

**查看slab的一些命令：**

1）awk '{print $1, $3\*$4/1024/1024"MB"}' /proc/slabinfo | sort -nr -k 2 -t' ' | head -n 10

按占用slab的大小列表

2）cat /proc/slabinfo

3）slabtop

## 优化内存分配策略（已合入）

将/proc/sys/vm/overcommit\_memory参数修改成1，允许分配所有的物理内存，而不管当前的内存状态如何，这个参数的意义如下：

|  |
| --- |
| overcommit\_memory:  This value contains a flag that enables memory overcommitment.  When this flag is 0, the kernel attempts to estimate the amount of free memory left when userspace requests more memory.  When this flag is 1, the kernel pretends there is always enough memory until it actually runs out.  When this flag is 2, the kernel uses a "never overcommit" policy that attempts to prevent any overcommit of memory.  This feature can be very useful because there are a lot of programs that malloc() huge amounts of memory "just-in-case" and don't use much of it.  The default value is 0. |

这个方法试验结果如下：

部署虚拟机前执行了：echo 1 > /proc/sys/vm/overcommit\_memory，然后部署60G的虚拟机，部署成功。

部署前后内存对比信息如下;

部署前

[root@gdnaf-ps-imas-rcs-compute36 ~]# free -m -h

total used free shared buffers cached

Mem: 125G 73G 51G 4.1G 184M 3.5G

-/+ buffers/cache: 70G 55G

Swap: 2.0G 16K 2.0G

[root@gdnaf-ps-imas-rcs-compute36 ~]#

部署后

[root@gdnaf-ps-imas-rcs-compute36 ~]# free -m -h

total used free shared buffers cached

Mem: 125G 76G 49G 4.1G 187M 4.2G

-/+ buffers/cache: 72G 53G

Swap: 2.0G 16K 2.0G

## 调整Journal日志大小（已合入）

有的环境Shmem能占用好几个G，其中journal日志占用内存比较多，通过修改journal日志的配置文件可以调整journal日志的大小以及修改journal日志存放到硬盘，但是后者修改后，会影响性能，所以不可行。

目前版本中已经做了journal日志大小的修改，在/etc/systemd/journald.conf中加了如下配置：RuntimeMaxUse=500m。如果不配置的话，那么journal日志可以占用tmpfs文件系统映射大小的10%。

## 回收缓存区（根据需要手动操作）

随着运行时间的增长，cached和buffer空间会越来越大，这部分是可以回收部分。操作系统会根据内存情况进行自动回收。当然也可以根据自己的需要进行手动回收：

回收方法：执行sync和echo 3 > /proc/sys/vm/drop\_caches

可以在部署虚拟机的时候检查可用内存是否够用，如果不够则执行这个回收操作。

缺点：linux的缓存机制可以加快读写盘的速度，如果把这部分内存回收的话，可能会引起性能下降。

## 回收Hugepage空间（根据需要手动操作）

对于不需要用到巨页的环境，比如广州现场环境用的ovs，没有用到ovdk，所以HugePage可以回收，回收方法是修改grub参数：

1)修改/etc/default/grub

vi /etc/default/grub 去掉hugepage配置(可能好几处),去掉default\_hugepagesz=2M hugepagesz=2M hugepages=20480或 default\_hugepagesz=1G hugepagesz=1G hugepages=2

存盘退出。

2)然后执行：

grub2-mkconfig -o /boot/grub2/grub.cfg

3)再重启单板。

# 中断绑核问题：

## 查看当前中断运行CPU

首先检查irqbalance服务的状态  
systemctl status  irqbalance   
如果是 active (running) 状态则执行下面两条命令停止掉irqbalance服务之后再进行中断绑定  
systemctl stop irqbalance  
systemctl disable  irqbalance  
如果是 inactive (dead)态，则直接执行中断绑定。

然后是物理信息：

lscpu和

cat /proc/cpuinfo |grep "processor" 、cat /proc/cpuinfo |grep "core id" 和 cat /proc/cpuinfo |grep "physical id"

下面是查看绑核

1）cat /proc/interrupts   
查看哪些中断占用资源比较多，都绑定到0核,注意没有中断号的中断没有办法进行绑定

2）绑定命令：  
中断绑定的方法：echo 核掩码 > /proc/irq/中断号/smp\_affinity，  
比如将50号中断绑定到0核上：echo 0001 > /proc/irq/50/smp\_affinity；  
核掩码的计算方法：如果是n核，掩码就是2的n次方，之后转为16进制：如1核就是0002

如果是smp\_affinity\_list，直接写数字0

ps -aux 查看要绑定的线程的PID,如果将线程2133绑定到核8上执行taskset -p 100 2133

3）自动绑核脚本，例如要把中断都绑定到0核  
#!/bin/bash  
function do\_task  
{  
    irqs=`ls -l /proc/irq/ | awk '{print $9}'`  
    num1=`ls -l /proc/irq/ | grep root -c`  
    local i=1  
    local irq\_id=""  
    local affinity=""  
    while [ $i -le $num1 ]  
    do  
        irq\_id=`echo $irqs | awk '{print $'$i'}'`  
        if [ $irq\_id != 'default\_smp\_affinity' ]  
        then  
            affinity=`cat /proc/irq/$irq\_id/smp\_affinity`  
            if [ $affinity != '00000001' ]  
            then  
                echo 00000001 > /proc/irq/$irq\_id/smp\_affinity  
            fi  
        fi  
        let "i+=1"  
    done  
}  
do\_task

4）注，有一些中断不能被绑核，比如0中断，2中断等。

5）注意董利娜的脚本服务performance.service，这个服务start的时候会将中断平均分配到除了被核隔离之外的核上。

## 核隔离

修改cat /etc/default/grub，修改isolcpus=4-13

修改完成后记得执行grub2-mkconfig -o /boot/grub2/grub.cfg后再重启系统才能生效。  
重启之后可以通过/proc/cmdline的内容确定修改是否成功。

# vg，lv挂载扩容问题：

## 检测命令

1）lsblk查看各个盘上挂载的vg和lv

2）vgs和vgdisplay查看有哪些vg占了多大空间

3）pvdisplay查看pv和所属vg

4）lvs和lvdisplay查看lv路径，lv名，所属vg名

5）lsof /dev/mapper/vg\_sys-lv\_root查看有没有进程占用某个lv

6）cat /proc/cmdline查看有没有用到lv

## 删除和扩容问题

**例一：第二块硬盘sdb为vg\_data卷，加到根分区**

1）先用上面的命令检测vg\_data和上面的lv有没有被使用

2）lvremove，vgremove，查看

3）扩容，vgextend vg\_sys /dev/sdb1，即将/dev/sdb1划为vg\_sys卷组，这时vgdisplay的空间已经改变，但df –h里的没有改变，lsblk的sdb1下也没有挂载

4）lvextend –l +100%FREE /dev/mapper/vg\_sys-lv\_root（剩下的空间都给lv\_root，-L指定G）

5）vgchange –a y vg\_sys 激活

6）resize2fs /dev/mapper/vg\_sys-lv\_root重新mount

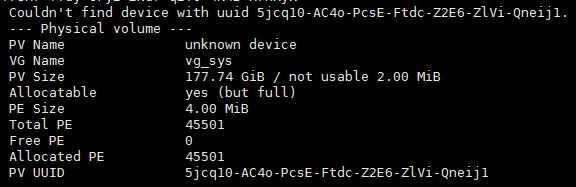
可能的问题：

1、创建分区的时候提示：WARNING: Duplicate VG name vg\_sys: Jl7X4O-qZlE-8bEe-BcU8-nCWj-HqYJ-zIzvje (created here) takes precedence over TT78Ar-fYdy-cFy2-ENdr-q2YO-4K4B-RrnRyx

类似这样的打印。

解决：查看lvdisplay vgdisplay pvdisplay

假如有：



说明有vg但是没有对应的pv。

1）执行vgreduce --removemissing vg\_sys去掉vg\_sys没有pv的部分。

2）然后把sdb1上的vg卷改名删除，在vgs里找到对应sdb1盘vg的uuid，执行vgrename Jl7X4O-qZlE-8bEe-BcU8-nCWj-HqYJ-zIzvje vg\_data重命名为vg\_data，然后就可以删除这个vg了，这样之后sdb1上就没有vg了，可以按上面的步骤进行扩容。

**例二：/dev/sda5为vg\_data卷，用了lv\_swap=4G，剩余空间够大，希望加到根分区**

1）lsblk查看

2）swapoff –a，之后free看swap空间是否已经变为0（释放lv\_swap）

3）lvremove /dev/vg\_data/lv\_swap，删除swap分区（lvs确认）

4）vgremove vg\_data /dev/sda5删除sda5上的vg\_data（vgs确认）

5）pvremove /dev/sda5删除sda5上的PV（pvs确认）

6）删除分区，重新划分区(d是删除sda5，默认，n创建两个盘，默认sda5,8G,再创建一个6,用完剩余空间，w写入)

fdisk /dev/sda

d

n

+8G（划一个8G的空间给新vg\_data）

n

（剩余的都给sda6）

w

7）partprobe /dev/sda更新分区表（lsblk确认）

8）vgextend vg\_sys /dev/sda6（把sda6扩给vg\_sys）

9）lvextend -L +400GB -n /dev/vg\_sys/lv\_root（lv\_root根分区扩容+400G）

10）resize2fs /dev/vg\_sys/lv\_root（调整文件系统大小，df –h确认）

11）pvcreate /dev/sda5（创建sda5的PV）

12）vgcreate vg\_data /dev/sda5（sda5上创建vg\_data）

13) lvcreate -L 4G -n lv\_swap vg\_data(给vg\_data创建4G的lv\_swap)

14）mkswap /dev/vg\_data/lv\_swap（创建swap）

15）swapon –a（开启lv\_swap，free –h确认swap空间有4G）

最后要检查fstab各个挂载点是否仍然存在。

帖子：<http://dev.zte.com.cn/topic/#/19150>

## swap分区删除可能会导致进入dracut

帖子：<http://dev.zte.com.cn/topic/#/20172>

## pane is dead和valueError:name already in use

例：在anaconda阶段报pane is dead错。

一般是lv和vg没有清干净，可以进控制台ALT+F2，fdisk –l看能不能找到盘，然后用lvs，vgs查看有没有残留的分区。

用最新的ks文件启动，可以清干净（lvm\_clean()函数）。

如果还有报错，如：

valueError：name already in use

1)cd tmp

2)grep –ril lvm\_clean找一个ks开头的脚本，./ks。。。执行，再清除一次。

## 文件系统损坏修复

例：vg\_sys-lv\_root进入失败导致系统进入dracut

用dd命令制作centOS系统盘，重启服务器从U盘启动，选择故障定位，修复系统，选择continue将mount系统盘，选择skip跳过，直接进入shell。

执行fsck –y /dev/mapper/vg\_sys-lv\_root修复根分区

执行tune2fs –l /dev/mapper/vg\_sys-lv\_root检查文件系统，状态是clean表示修复成功。

注：1期环境出现过存储内容超过文件系统大小的情况，这时要先确认剩余空间，然后对vg\_sys-lv\_root扩容，再进行修复。

# 升级问题：

## 检查

# 查看cpu使用：

## 查看cpu设置

看cpu设置，lscpu

## 有虚机情况下CPU分析

看虚机进程：ps –A | grep qemu

看进程用的cpu：taskset –pc $pid

# 虚机实例问题：

## 查看虚机

在控制节点：

virsh list

. keystonerc\_admin或者source keystonerc\_admin

nova list

查看虚机和实例

然后用浏览器登陆控制节点IP进入控制台

## 起虚机

**PXE方式安装：**

1. 创建img文件.

qemu-img create -f qcow2 /home/vm/image/pxetest.img 100G

1. 创建网桥（如果机器只有一块网卡）

ovs-vsctl --no-wait add-br br\_ovs

1. 创建xml文件（这里创建的文件名为pxetfg.xml）

<domain type="kvm">

  <name>pxe</name>

  <memory>1242880</memory>

  <memoryBacking>

  </memoryBacking>

  <vcpu>4</vcpu>

  <os>

    <type>hvm</type>

    <boot dev="network"/>

  </os>

  <features>

    <acpi/>

    <apic/>

  </features>

  <clock offset="utc">

    <timer name="pit" tickpolicy="delay"/>

    <timer name="rtc" tickpolicy="catchup"/>

    <timer name="hpet" present="no"/>

  </clock>

  <cpu mode="host-model"/>

  <devices>

    <emulator>/usr/bin/qemu-system-x86\_64</emulator>

    <disk type="file" device="disk">

      <driver name="qemu" type="qcow2" cache="none"/>

      <source file="/home/vm/image/pxe.img"/>

      <target bus="virtio" dev="sda"/>

    </disk>

    <interface type='bridge'>

      <mac address='24:52:00:83:03:18'/>

      <source bridge='br\_ovs'/>

      <virtualport type='openvswitch'>

      </virtualport>

      <model type='virtio'/>

    </interface>

    <serial type="pty"/>

    <input type="tablet" bus="usb"/>

    <graphics type="vnc" autoport="yes" keymap="en-us" listen="0.0.0.0"/>

    <video>

      <model type="cirrus"/>

    </video>

  </devices>

</domain>

上面的xml文件中红色字体部分是根据自己的具体需求来配置的。

1. 创建pxe服务器

./ZXTFG-1.0-1.KVM.CentOS-7.1.x86\_64.bin

注意创建pxe服务器时，网口名需要写成刚才网桥创建的名字，即:br\_ovs，其它参数按照需求自行配置。

1. 创建虚拟机

virsh create pxetfg.xml

后面的pxetfg.xml是刚创建的文件。

1. 安装完成之后删除虚拟机

virsh destroy pxe

后面的pxe这个名字和xml文件中<name>字段对应，也可以使用如下命令查看：

virsh list

1. 修改xml文件，设置从硬盘启动

将上面的xml文件中的os部分改为如下：

  <os>

    <type>hvm</type>

    <boot dev="hd"/>

  </os>

1. 重新创建虚拟机

virsh create pxetfg.xml

**ISO方式安装：**

在使用光盘安装时，不能直接使用target目录下的iso文件，因为这个iso是直接从os目录下拷贝过来的，可就是CentOS Minimal的原版镜像，里面没有ks文件，也没有我们编译的rpm包，因此需要重新编译ISO，具体如下：

./ ZXTFG-1.0-1.KVM.CentOS-7.1.x86\_64.bin mkiso

执行上面的命令之后，会在当前目录下重新生成一个iso文件。这个iso就是我们需要的ISO文件。

目前光盘安装验证主要是通过虚拟机来完成的。iso的安装和pxe类似，只需要修改下xml配置文件即可。

需要将启动方式设置为光驱启动，并添加一个disk，device为cdrom，具体修改部分如下：

<os>

    <type>hvm</type>

    <boot dev="cdrom"/>

  </os>

<disk type="file" device="cdrom">

<source file="/home/vm/iso/ZXTFG-1.0-1.KVM.CentOS-7.1.x86\_64.iso"/>

<target bus="ide" dev="hdb"/>

</disk>

# daisy问题：

## 流程

daisy部署流程及可能的问题：

服务器：

1) dhcp/nfs->http/tftp服务提供

2）pxe\_server\_install data.json

命令作用：这个命令的主要作用是安装dhcp服务器、tftp服务器、nfs服务器；生成/tftpboot、/linuxinstall、/home/install\_share三个共享文件夹；在tftp路径下准备好目标机需要的default文件、内核文件、可以跑discovery脚本的最小linux系统initrd.img。

入参说明：json格式的配置文件，主要是涉及服务器本身的一些配置，包括DHCP监听端口、监听IP、DHCP服务器分配ip的范围等，例子如下：



备注：在DHCP监听配置固定好的情况下，这个命令在服务器上只需要执行一次。

3)ps\_os\_install os.json

入参说明：json格式的配置文件，涉及目标机的一些配置，包括目标机的root密码、安装盘大小、内存大小、各个分区大小、巨页、主机名、各个网口的配置、网口固化配置、网口bond、目标机安装系统iso文件路径、目标机上和服务器连接的网口mac地址、目标机安装完是否重启等配置，其中大部分配置都是discovery脚本获取到的。例子如下：



返回值说明：返回0 表示成功

返回1表示失败，这个时候在打印中可以看到[error]开头的提示行表示失败的原因。

命令作用：这个命令主要作用是按照入参给的那些配置生成目标机安装需要的kickstart文件；在3个共享文件夹下准备好目标机需要的default文件（以mac命名）、内核文件、正常的最小linux系统initrd.img、目标机要安装的操作系统iso、kickstart文件以及其他一些用户文件。

备注：每安装一个目标机，这个命令需要执行一次，入参json文件里的关键字值不一样。

目标机：

1）bmc网口启动ipmi总线，通过dhcp要ip，

2）在/etc/dhcp/dhcpd.conf看设置，哪里下载pxelinux.0

3）通过网络下载pxelinux.0文件到tftpboot/pxelinux.0中启动

4）根据这个文件下载default（先mac，后default搜到就不搜了），99.99.1.5下的路径

5）在default里有kernel和小系统initrd.img的路径，ks路径，通过tftp下载，在/tftpboot/

6）启动小系统，主机发现

7）通过ironic\_discovery收集上电启动需要的信息，服务器ironic放入数据库

8）服务器收到后上报给daisy，daisy获取后给客户对os目标机进行配置，下发给ironic，得到json文件，ironic调用pxe\_os\_install根据配置生成目标机要的ks文件，这里包括主机名、磁盘分区、网口固化、ip配置、bond配置等。

9）建立目标机的mac目录，/tftpboot/路径下面，里面放了iso，需要的ks文件，正常的initrd.img

9）小系统重启，下载安装正常小系统，通过pxe安装。

10）目标机安装完，进度问题，看kvm。

详细：D:\SVN检出\linux积累\06\_OS\pxe安装工具实现\pxe\_install\_tool\doc

## 理解

1）daisy传os.json文件，有glance\_lv\_size就会创建lv\_glance，有mount glance=yes就会mount到系统，在/dev/mapper下能查看有没有lv\_glance，在/dev/vg\_data/lv\_glance下看有软链。用daisy部署的节点才有glance，否则json文件里配了没用。

## 问题

1）DHCP阶段

首先看服务器的DHCP服务状态

查看nfs服务，systemctl status nfs

如果是容器，则docker-manage enter daisy查看docker进程和进入daisy

. /root/daisyrc\_admin显示节点

检查/tftpboot下的文件

ps -aux | grep dhcp查看dhcp进程

cat /var/log/pxe\_install.log搜索服务器安装信息

网口/etc/syscfg/network-scripts/ifcfg

有没有作bounding网桥

cd /var/lib/dhcpd

tcpdump –I 网口 port67/68 dhcpd.leases看mac地址和租约时间

这里data.json文件里有网口和mac地址。

journalctl日志可以看到dhcp服务信息。

/tftpboot/pxelinux.cfg/default目录01-mac目录

2）进度问题

3) docker管理查看json：

tail /var/log/pxe\_install.log

ll /linuxinstall/74-4a-a4-00-d6-92/

tail /var/log/pxe\_os\_table.log

docker-manage ps

docker-manage enter daisy

ll /home/install\_share/74-4a-a4-00-d6-92/

cd /home/install\_share/74-4a-a4-00-d6-92/

cat /etc/exports

cat os.json |grep disk

docker-manage ps

docker-manage enter daisy查看docker进程和进入daisy

# 升级opencos问题：

## 对应版本

1.0-CGSL\_V5 -> opencos

2.0-Mimosa -> TECS

对应版本的才能升级

## 升级日志

1）看opencosos.log对应opencos系统

2）看cat /var/log/tfg/tfg-upgrade.log对应tfg系统

## 问题

1.opencos升级打印有[ FAIL ] superpath install error

问题是环境中有更高版本的superpath，在opencosos升级日志中能看到已经存在有最新版本的superpath，已经升级成功，但是会打印这个错误。

2.docker-tmanage已经变成docker-manage，如果存在则会显示升级失败日志。

3.部分升级失败的包可以先尝试手动安装。

# 网口问题：

## ifconfig看不到网口

1）lspci | grep Eth 查找网口，查看数量，种类

[root@bsp64 ~]# lspci | grep Eth

81:00.0 Ethernet controller: Intel Corporation I350 Gigabit Backplane Connection (rev 01)

81:00.1 Ethernet controller: Intel Corporation I350 Gigabit Backplane Connection (rev 01)

2）cat /etc/udev/rules.d/71-nicfix.rules查看网口名和对应的网卡

3）ethtool -i eth0(网口名)，可以看到驱动，版本

4）比较uname –a 和modinfo里内核和驱动版本，或者rpm -qa igb/kernel看信息

5）rmmod igb（驱动名）卸载驱动

6）modprobe igb(驱动名)，加载驱动命令，没有打印表示成功，可以查看驱动加载失败，或者dmesg找驱动异常打印。

7）depmod `uname -r`重新生成ll

/lib/modules/3.10.0-229.1.el7.x86\_64/modules.dep

8）ixgbe的配置参数在cat /etc/modprobe.d/tecs-sriov.conf

9）如1）中所示的网口信息可以lspci –vmmks 81:00.0查看这一个网口的信息，节点、设备、驱动、模块等，这里igb\_uio表示做了托管，托管的网口ifconfig看不到。

## 控制节点ping磁阵丢包，计算节点正常

例：控制节点10.43.78.47，磁阵192.168.64.247

1）可能的问题，存在相同的IP，检查ip addr |grep 192

2）存在相同的mac地址，检查ifconfig |grep ether

3）route查看网关

4）traceroute 192.168.64.247追踪包，（需要traceroute的rpm包）

5）cat /etc/sysconfig/network-scripts/ifcfg-enp129s0f0.807查看配置文件

6）cat /etc/resolv.conf查看

7）抓取指定网口发出和接收的包tcpdump -i enp129s0f0.807

8）检查缓存，ethtool –g eth0，其中TX/RX当前硬件设置，缓存大小。

修改文件

vim  /etc/rc.local

ethtool -G eth0 tx 2048

ethtool -G eth0 rx 2048

9）arq –n | grep 192.168 查看arq，arq请求mac地址有没有成功。

这个问题中，ping 247丢包，247的网口是enp129s0f0.807，发现有部分返回包发到了enp129s0f0.806网口，造成丢包，检查下中间的交换配置，还有确认下是不是两个控制节点和计算节点3,4在一个机框里。

## tcpdump抓包

例：

1）tcpdump --e host ICE（主机名ICE，或IP）（<表示接收，>表示发送）

2）tcpdump host 210.27.48.1（接收所有主机发出和收到的分组）

3）tcpdump ip host 210.27.48.1 and ! 210.27.48.2（除主机210.27.48.2的包，ip包，可以\（210.27.48.2 or 210.27.48.3\））

4）tcpdump tcp port 23 and host 210.27.48.1

-i 指定监听的网络接口

-e打印数据链路头部信息

## 分配ip后ping不通

分配IP

1.dhclient

2.ifconfig eth0 192.168.1.100  netmask 255.255.255.0即时生效

3.vim  /etc/sysconfig/network-scripts/ifcfg-eth0  
       --增加或者修改下面几行    
        BOOTPROTO=static  
        IPADDR=192.168.1.100  
        NETMASK=255.255.255.0重启生效

route -n查看网关

修改默认网关

1.route add default gw 192.168.1.1即时生效(del删除老的)

2.vim /etc/sysconfig/network  
         --增加或修改下面一行  
           GATEWAY=192.168.1.1重启生效

修改DNS  
重启生效：vim /etc/resolv.conf  
          --增加下面一行  
          nameserver 202.96.134.133

## 网口固化问题

正常环境作网口固化，固化网口为固定的名字，ip，等。

1、udev会探测硬件变化，在以下目录生成规则文件：

/etc/udev/rules.d/71-nicfix.rules

规则设置为如下形式：

SUBSYSTEM=="net", ACTION=="add", KERNELS=="0000:13:00.0", NAME="ebl"

SUBSYSTEM=="net", ACTION=="add", KERNELS=="0000:13:00.1", NAME="ebr"

2、在

/etc/sysconfig/network-scripts/目录，会生成文件ifcfg-ebl：

DEVICE="enl"

BOOTPROTO="static"

HWADDR="20:12:06:11:05:50"

NM\_CONTROLLED="no"

ONBOOT="yes"

TYPE="Ethernet"

NETMASK=255.255.254.0

GATEWAY=10.43.166.1

IPADDR=10.43.166.55

这里要求配置好DEVICE，BOOTPROTO，物理地址，ONBOOT，掩码网关IP等。

3、在ifconfig和ip addr下就可以看到对应网口的信息。

例：环境网口固化失败问题的分析：

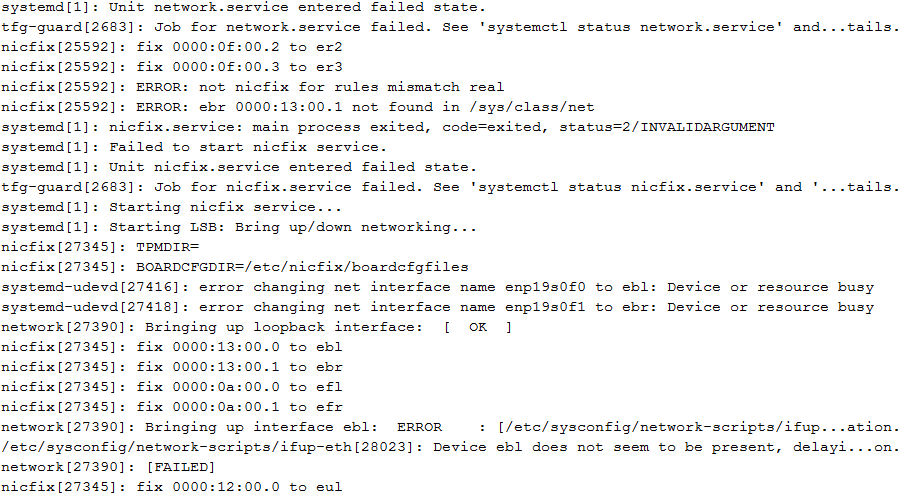
a.先看一下上述文件的配置有没有问题

b.systemctl status network.service错误信息提示。

c.dmesg中搜错误信息或网口信息。

d. udevadm trigger --action=add --subsystem-match=net命令重新触发网口固化。然后用journalctl –f跟踪看一下出错。

e.根据journal信息，可以查到网口固化的记录：



比如这样的，就可以查nicfix服务，/sys/class/net这个目录，Device or resource busy，以及fix \*\* to ebl这样的记录。

f.例中看到device busy打印，则要先将网口退出，ifdown enp19s0f0和1，再运行命令进行重新触发网口固化，成功修复。做网口修改操作，务必要将网口down掉之后再开启。

另：附/sys/class/net这个目录中的内容：



其中网口固化的efl这样的文件，其实是一个到对应pci下文件的软链。

这些文件中的信息是会随着对硬件的探测而改变的。

附：net文件夹下全文件内容解析：

<https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-class-net>

再：Udev的运行规则：

1、系统起来后，内核对网口的命名以eth(x)来命名，到UDEV阶段会根据新规则，结合位置重新命名。  
2、UDEV的工作机制，当产生一个设备add事件时，UDEV会创建一个任务去实现网口固化工作，因为存在多个网口  
那么UDEV就会创建多个任务，多个任务是并发运行，谁先谁后不确定。如果我们想命名的eth0/eth1/eth2/eth3占用，  
那么就会命名失败，见上面的打印信息。如果“renamed network interface eth1 to enp4s16f1”在“ eth64 to eth1”之前就不会有问题

帖子：<http://dev.zte.com.cn/topic/#/9305>

## 查看网线连接状态

1、ethtool + 网口名，看Link detected这一栏，yes表示连接了网线，no表示没有连接。

2、ifconfig –a看RUNNING的网口

## 查看网卡mac地址：

1. ifconfig -a 其中 HWaddr字段就是mac地址  
2. cat /sys/class/net/eth0/address 查看eth0的mac地址  
3. cat /proc/net/arp 查看连接到本机的远端ip的mac地址  
4. 程序中使用SIOCGIFHWADDR的ioctl命令获取mac地址

# dhcp：

## 检查dhcp服务状态和连接情况

yum -y install nmap

nmap --script broadcast-dhcp-discover

dhcp报文有4种，discover，offer，

# 配外网：

**1）设置linux代理**

将下面脚本解压拷贝到/etc/profile.d，然后重新登录或执行"source /etc/profile.d/net\_proxy.sh"

**脚本内容：**

declare -x http\_proxy="proxynj.zte.com.cn:80"

declare -x https\_proxy=$http\_proxy

declare -x ftp\_proxy=$http\_proxy

declare -x no\_proxy="10.0.0.0/8,localhost,127.0.0.1,.zte.com.cn"

**2）设置dns服务器**

echo nameserver 10.30.1.9 > /etc/resolv.conf

**3）在公司内网访问外网时，可以下载下面链接脚本，解压后执行，也可以使用firefox登录一下**

python login.py   
User account: 10067372

Password:

Login Success. Enjoy your surf! (^.^)

**脚本内容：**

#!/usr/bin/env python

import httplib, urllib,ssl

import sys, os

from urlparse import urlparse

from getpass import getpass

httplib.HTTPConnection.debuglevel = 100

marker = "http://www.baidu.com/"

class bcolors:

HEADER = '\033[95m'

OKBLUE = '\033[94m'

OKGREEN = '\033[92m'

WARNING = '\033[93m'

FAIL = '\033[91m'

ENDC = '\033[0m'

BOLD = '\033[1m'

UNDERLINE = '\033[4m'

def lookup\_proxy\_auth\_ip(proxy):

conn = httplib.HTTPConnection(proxy)

conn.request("GET", marker)

response = conn.getresponse()

conn.close()

if response.status == httplib.FOUND:

for item in response.getheaders():

if item[0]=='location':

return response.status, item[1]

return response.status, None

proxy = os.getenv("http\_proxy") or os.getenv("HTTP\_PROXY") or raw\_input("Proxy sever: ")

proxy = urlparse(proxy).netloc or proxy

status, auth\_url = lookup\_proxy\_auth\_ip(proxy)

if status == httplib.OK:

print bcolors.OKGREEN + "You already can access %s" % marker+bcolors.ENDC

sys.exit(0)

account=raw\_input("User account: ")

passwd=getpass()

print auth\_url

auth\_server = urlparse(auth\_url).netloc

scheme = urlparse(auth\_url).scheme

'''

opr=pwdLogin&userName=10092495&pwd=xxxx&rememberPwd=0&lang=chs

'''

params = urllib.urlencode({

"lang": "chs",

"opr" : "pwdLogin",

"pwd" : passwd,

"rememberPwd" : "0",

"userName" : account

})

headers = {"Content-Type": "application/x-www-form-urlencoded"}

if (scheme == 'http'):

conn = httplib.HTTPConnection(auth\_server)

else:

conn = httplib.HTTPSConnection(auth\_server, context=ssl.\_create\_unverified\_context())

conn.request("POST", "/ac\_portal/login.php", params, headers)

response = conn.getresponse()

conn.close()

if response.status == httplib.OK:

print bcolors.OKGREEN+"Login Success. Enjoy your surf! (^.^)"+bcolors.ENDC

# bin问题：

## 修改和重新生成bin包

1）解压bin包

./\*.bin --noexec --target bin

即将bin包解压到bin这个目录

2）修改完后制作bin包

/tfg/tools/release/makeself/makeself.sh bin/

Mimosa-V02.16.11.P7B1I60-CGSL\_VPLAT-5.1-x86\_64-KVM.bin "Mimosa" ./setup/setup.sh

# iso问题：

## 修改和重新打iso包

tar -cf - ./ | tar -xf - -C ${PACKDIR} //在mount iso后的目录中执行，${PACKDIR}是将要打包的iso的目录 (目的：从mount路径拷贝出来，到${packdir})

mkisofs -R -T -r -V "CD-CGSL\_VPLAT" -l -d -allow-multidot -allow-leading-dots -no-bak -o ${CGS\_ISO\_FILE} -b isolinux/isolinux.bin -c isolinux/boot.cat -no-emul-boot -boot-load-size 4 -boot-info-table ${PACKDIR}

然后执行bin，重新生成iso。（5: make iso install media.）

# syslinux问题：

## exec syslinux cmd on … failed

syslinux的路径是写死的，核查是否我们的操作系统，

1）uname -a

2）rpm –qa | grep tfg

3）cd /var/tfg-sys/tfg-depend

# 进程问题：

## 查找进程号，查父进程，看堆栈等

1）top，ps aux 查看当前进程

top可以按M根据内存排序，ps aux | grep进程，ps aux –sort=-rss | head –n 20

cpu loaderaverage是平均负荷，top还可以看cpu数

2）查找进程文件，locate 或find –name= 或rpm –qf 或rpm –q –whatprovides看文件属于哪个包

3）cat /proc/$pid/status看详细进程状态，可以查父进程PPID

4）cat /proc/$pid/task/$pid/stack看堆栈

## 进程状态的含义



## 进程挂死

例：运行make iso install失败，程序挂死。

1）找到打印挂死最近的命令，rpm –qa，手动运行发现挂死。

2）查找rpm的进程，发现了多个后缀，/var/lib/rpm/.dbenv.lock的进程，为低优先级。

3）strace追踪rpm，strace rpm –qa，发现卡在，FUTEX\_WAIT，判断是由于死锁。

4）lslocks查看锁的状态，能看到pid，进程名和锁文件

5）杀完lock进程后，执行命令还是挂死，又重新生成了锁。

6）查看这些锁的父进程是PackageKit，判断是装了这个和yum有冲突，导致rpm数据库损坏。

7）修复数据库：

rpm -v --rebuilddb

# 启用samba服务：

1、安装samba的rpm包，

samba-4.1.1-38.el7.x86\_64

samba-libs-4.1.1-38.el7.x86\_64

samba-common-4.1.1-38.el7.x86\_64

samba-client-4.1.1-38.el7.x86\_64

2、设置samba服务器

默认设置文件/etc/samba/smb.conf

例：创建samba共享目录

[home]

comment = centos all file

path = /

valid users = root

public = no

writable = yes

printable = no

create mask = 0777

directory mask = 0777

3、smbpasswd -a username创建一个带密码用户

注：/etc/samba/smb.conf文件修改后，几分钟后自动重新加载，和手动一样。

# rpm命令被删除和数据库修复：

## rpm命令被删除

准备正常的文件1./bin/rpm，2./usr/lib/下的so文件，3./usr/lib/rpm文件夹

一、把/bin/rpm放到/bin目录下，linux终端下输入rpm命令，出现help为存在此命令。

如果出现以下类似错误：需要进行第二步骤

rpm: error while loading shared libraries: librpm-4.4.so: cannot open

shared object file: No such file or directory

二、把文件2放到./usr/lib/下，

使用[root@localhost home]#rpm -qa，检查某个rpm是否安装，如果出现

error: cannot open Packages database in /var/lib/rpm

no packages

使用[root@localhost home]#rpm --rebuilddb，修复rpm数据库，如果出现

error: unrecognized db option: “db3″ ignored.

error: cannot open Packages index

出现以上情况，说明文件没有copy完，进行第三步骤

三、把backusrlibrpm.zip解压到./usr/lib/下，并确保rpmpopt-4.1,rpmi,rpmv等文件的存在。

再次运行rpm安装文件时报错:

rpmdb: unable to join the environment　　  
　　error: db4 error(11) from dbenv->open: Resource temporarily unavailable　　  
　　error: cannot open Packages index using db3 - Resource temporarily unavailable (11)　　  
　　error: cannot open Packages database in /var/lib/rpm　　  
　　no packages

解决方法一：是执行“export LD\_ASSUME\_KERNEL =2.2.25”或“export LD\_ASSUME\_KERNEL =2.2.25”命令，也可以将其写入/etc/bashrc。  
  
解决方法二：是执行“export LD\_ASSUME\_KERNEL =2.2.25”或“export LD\_ASSUME\_KERNEL =2.2.25”命令，升级rpm到rpm-4.2-1.i386.rpm

解决方法三：是执行“export LD\_ASSUME\_KERNEL=2.2.5”

## rpm数据库恢复

1）有的时候,我们的rpm的数据库会因为某些原因损坏,这样当你执行rpm的相关命令的时候就会报类似下面的错误:  
rpmdb: fatal region error detected; run recoveryerror: db4 error(-30982) from dbenv->open: DB\_RUNRECOVERY: Fatal error, run database recoveryerror: cannot open Packages index  
这样的后果是你将不能使用rpm来添加删除,更新查询你系统的RPM软件包了,其实只要按照下面的步骤做就能恢复你的RPM数据库而不必重新安装系统  
# cd /var/lib# mv rpm rpm.old# mkdir rpm# rpmdb --initdb# cp rpm.old/Packages rpm# rpmdb –rebuilddb（老版本rpmdb命令就是rpm）

2）当进行包安装、升级、卸载等操作时，可能损坏RPM DB，这时需要通过--rebuild参数来修复RPM DB，但有时会碰到这样的问题：  
[root@tiv125 lib]# rpm -v --rebuilddb  
rpmdb: fatal region error detected; run recovery  
error: db4 error(-30982) from dbenv->open: DB\_RUNRECOVERY: Fatal error, run database recovery  
error: cannot open Packages index  
这时需要先手动删除/var/lib/rpm目录下的 \_\_db.\* 文件，比如 \_\_db.001, \_\_db.002, etc.  
然后再运行 rpm -v --rebuilddb 即可。

<http://www.cnblogs.com/hakuci/archive/2008/12/07/1349592.html>

# audit监控文件夹修改：

## 例/var/log/httpd文件夹被删除导致升级tecs失败

手动创建文件夹可以升级成功，为规避，监控这个目录。

**1、部署操作： auditctl -w /var/log/httpd -p w -k httpd**

   说明： -w /var/log/httpd 表示对这个文件夹进行监控

          -p w 表示监控目标的写操作，如果读、写、改变属性等操作都监控，那么就是-p rwa

          -k httpd 设置一个关键字，方便查找

   注意： 用这个命令创建的规则，重启不会生效，为了让这个规则重启也生效，最好是加入规则文件

**vi /etc/audit/rules.d/audit.rules**

          在最后加入一行**-w /var/log/httpd -p w -k httpd**

**2、获取监控结果，有两种方法**

   1）cat /var/log/audit/audit.log |grep httpd

   2）ausearch -k httpd

   httpd就是部署监控时设置的关键字，建议使用第二种方法，第一种方法得到的时间是一个相对时间，而第二个方法是绝对时间，可读性强。

# 报device错误：

## 例device-mapper:table:253.:14:spp:error getting decive

1）先ll /dev看dev下的目录名，和table号对应的device

2）在ll /dev/mapper下看硬件到device的软链

3）如果没有这个设备，则可能是需要屏蔽multipath本地盘：

1、确定本地盘

[root@host-192-41-0-12 ~]# cat /sys/block/sda/device/vendor

LSI

[root@host-192-41-0-12 ~]# mount

proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)

sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)

devtmpfs on /dev type devtmpfs (rw,nosuid,size=38719860k,nr\_inodes=9679965,mode=755)

securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)

tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)

devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)

tmpfs on /run type tmpfs (rw,nosuid,nodev,mode=755)

tmpfs on /sys/fs/cgroup type tmpfs (rw,nosuid,nodev,noexec,mode=755)

cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,release\_agent=/usr/lib/systemd/systemd-cgroups-agent,name=systemd)

pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)

cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)

cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpuacct,cpu)

cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)

cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)

cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)

cgroup on /sys/fs/cgroup/net\_cls type cgroup (rw,nosuid,nodev,noexec,relatime,net\_cls)

cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)

cgroup on /sys/fs/cgroup/perf\_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf\_event)

cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)

configfs on /sys/kernel/config type configfs (rw,relatime)

/dev/mapper/vg\_sys-lv\_root on / type ext4 (rw,relatime,data=ordered)

systemd-1 on /proc/sys/fs/binfmt\_misc type autofs (rw,relatime,fd=37,pgrp=1,timeout=300,minproto=5,maxproto=5,direct)

mqueue on /dev/mqueue type mqueue (rw,relatime)

debugfs on /sys/kernel/debug type debugfs (rw,relatime)

sunrpc on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw,relatime)

binfmt\_misc on /proc/sys/fs/binfmt\_misc type binfmt\_misc (rw,relatime)

sunrpc on /proc/fs/nfsd type nfsd (rw,relatime)

/dev/sda2 on /boot type ext3 (rw,relatime,stripe=4,data=ordered)

/dev/mapper/vg\_data-lv\_mongodb on /var/lib/mongodb type ext4 (rw,relatime,data=ordered)

hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime)

2、获取本地盘的wwid

[root@host-192-41-0-12 ~]# /lib/udev/scsi\_id -g -u -d /dev/sda

3600508e000000000fa7248e5204eb005

[root@host-192-41-0-12 ~]#

3、将本地盘加入黑名单（具体查看修改的blacklist部分内容）

[root@host-192-41-0-12 ~]# vi /etc/multipath.conf

找到以下内容，把本机的wwid加进去

blacklist

{

wwid 3600508e000000000fa7248e5204eb005

}

4、重启服务

[root@host-192-41-0-12 ~]# systemctl restart multipathd

5、再次确认

[root@host-192-41-0-12 ~]# systemctl status multipathd.service

multipathd.service - Device-Mapper multipath Device Controller

Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled)

Active: active (running) since Tue 2016-07-19 11:17:11 CST; 6s ago

Process: 8394 ExecStopPost=/etc/rc.d/boot.multipath postStop (code=exited, status=0/SUCCESS)

Process: 8415 ExecStart=/sbin/multipathd (code=exited, status=0/SUCCESS)

Process: 8400 ExecStartPre=/etc/rc.d/boot.multipath preStart (code=exited, status=0/SUCCESS)

Main PID: 8418 (multipathd)

CGroup: /system.slice/multipathd.service

└─8418 /sbin/multipathd

Jul 19 11:17:11 host-192-41-0-12 systemd[1]: Starting Device-Mapper multipath Device Controller...

Jul 19 11:17:11 host-192-41-0-12 boot.multipath[8400]: load multipath load dm modules

Jul 19 11:17:11 host-192-41-0-12 boot.multipath[8400]: multipath load dm devices

Jul 19 11:17:11 host-192-41-0-12 boot.multipath[8400]: multipath load dm devices parts

Jul 19 11:17:11 host-192-41-0-12 systemd[1]: PID file /var/run/multipathd.pid not readable (yet?) after start.

Jul 19 11:17:11 host-192-41-0-12 multipathd[8418]: enter log\_thread\_start.

Jul 19 11:17:11 host-192-41-0-12 systemd[1]: Started Device-Mapper multipath Device Controller.

6、错误打印不再出现

# Linux编译源码包：

**方法1：**

假设该文件已经存在于/usr/local/src/下。在该目录下执行rpmbuild --rebuild --clean setarch-1.3-1.src.rpm

若顺利执行成功则会在/usr/src/redhat/RPMS/i386/目录下生成setarch-1.3-1.i386.rpm这个rpm文件。

**方法2：**

（1）rpm –ivh setarch-1.3-1.src.rpm   执行rpm安装命令

（2）cd /usr/src/redhat/SPECS 切换目录到/usr/src/redhat/SPECS （src.rpm包默认的解压目录）

（3）rpmbuild –bp setarch-1.3-1e.spec 执行rpmbuild会生成源码包

（4）cd /usr/src/redhat/BUILD/setarch-1.3-1   切换到生成的源码包

（5）./configure 编译配置

（6）make 编译

（7）make install 安装

**方法3：**

（1）rpm –ivh setarch-1.3-1.src.rpm

（2）cd /usr/src/redhat/SPECS

（3）rpmbuild –bb setarch-1.3-1.spec

执行rpmbuild操作，会在/usr/src/redhat/RPMS/i386（不同的包，产生的路径不相同，可能会是i686、noarch等）下创建一个或多个的rpm包

（4）rpm –ivh /usr/src/redhat/RPMS/i386/setarch-1.3-1.rpm

 注：当执行rpmbuild –bb setarch-1.3-1.spec 出现错误时，例：Error: Architecture is not included : i386，此错误表示该软件包不支持i386平台（默认的rpmbuild为i386平台），需指定到别的平台，指定参数为--target=i686

rpmbuild –bb –target=i686 setarch-1.3-1.spec

i386软件包可以在任何x86平台下使用，无论是i686还是x86\_64的机器；而i686的软件包一般都对cpu进行了优化，具有向后的兼容性，不具有向前的兼容性。

# 提取rpm包spec文件，分解源码包：

**方法1：**

rpm -i MySQL-5.1.31-0.glibc23.src.rpm  
使用该命令后，rpm会在/usr/src/packages/SOURCES/目录下生成mysql-5.1.31.tar.gz这个源代码包，这个文件也是我们希望得到 的最终文件。

同时rpm也会在/usr/src/packages/SPECS/目录下生成一个名为mysql-5.1.31.spec的文本文件，可以使用rpmbuild -bp mysql- 5.1.31.spec命令来构建mysqld。

**方法2：**

rpm2cpio MySQL-5.1.31-0.glibc23.src.rpm | cpio -idv  
该命令可以在不安装的情况下提取出rpm包中的文件，也可以使用其他类似命令：

rpm2cpio MySQL-5.1.31-0.glibc23.src.rpm | cpio -vi

rpm2cpio MySQL-5.1.31-0.glibc23.src.rpm | cpio -idmv

rpm2cpio MySQL-5.1.31-0.glibc23.src.rpm | cpio --extract --make-directories

参数i和extract相同，表示提取文件。  
v表示指示执行进程。  
d和make-directory相同，表示根据包中文件原来的路径建立目录。   
m表示保持文件的更新时间。

# 开机挂死问题：

## 查找进程号，查父进程，看堆栈等

1）top，ps aux 查看当前进程

top可以按M根据内存排序，ps aux | grep进程，ps aux –sort=-rss | head –n 20

cpu loaderaverage是平均负荷，top还可以看cpu数

# 修改环境变量，比如代理：

## yum源不能连接内网，HTTP302错误

方法：修改环境变量，添加

export no\_proxy="\*.zte.com.cni\*,10.0.0.0/8,10.43.0.0/16,10.43.177.140,10.43.166.102,127.0.0.1,10.43.166.59,10.43.166.55"

按变量的生存周期来划分，Linux变量可分为两类，它们的修改方法如下：  
（1）永久的：需要修改配置文件，变量永久生效。  
常见的配置文件包括：  
（1-1）/etc/profile：对所有用户生效；此文件为系统的每个用户设置环境信息,当用户第一次登录时,该文件被执行；并从/etc/profile.d目录的配置文件中搜集shell的设置  
例如：编辑/etc/profile文件，添加CLASSPATH变量  
# vi /etc/profile  
添加一行：  
export CLASSPATH=./JAVA\_HOME/lib;$JAVA\_HOME/jre/lib  
修改后需要执行重新登录才能生效，也可以执行命令source /etc/profile来生效  
  
（1-2）/etc/bashrc：对所有用户生效；为每一个运行bash shell的用户执行此文件.当bash shell被打开时,该文件被读取  
编辑方法如上，不再赘述  
  
（1-3）~/.bash\_profile：仅会对当前用户有效；每个用户都可使用该文件输入专用于自己使用的shell信息,当用户登录时,该文件仅仅执行一次  
例如：编辑guok用户目录(/home/guok)下的.bash\_profile  
$ vi /home/guok/.bash.profile  
添加如下内容：  
export CLASSPATH=./JAVA\_HOME/lib;$JAVA\_HOME/jre/lib  
修改后需要执行重新登录才能生效，也可以执行命令source /etc/profile来生效  
  
（1-4）~/.bashrc：仅会对当前用户有效；该文件包含专用于你的bash shell的bash信息,当登录时以及每次打开新的shell时,该该文件被读取  
编辑方法如上，不再赘述  
  
另外，~/.bashrc等中设定的变量(局部)只能继承/etc/profile中的变量,他们是"父子"关系  
  
综述，对上述文件修改，添加你需要的变量，在启动一个shell（终端，terminal）时，你所定义的变量均会生效的。  
  
（2）临时的：使用export命令声明即可，变量只在当前的shell(BASH)或其子shell(BASH)下是有效的,在关闭shell后失效，再打开新shell时就没有这个变量，需要使用的话还需要重新定义  
在shell的命令行下直接使用[export 变量名=变量值] 定义变量  
  
环境变量的查看  
（1）使用echo命令查看单个环境变量。例如：  
echo $PATH  
（2）使用env查看所有环境变量。例如：  
env  
（3）使用set查看所有本地定义的环境变量。例如：  
set  
另外，unset可以删除指定的环境变量。  
（4）环境变量修改

export http\_proxy=proxy\_addr:port

yum的话则可以单独设置，在/etc/yum.conf里面增加proxy=proxy\_addr:port

常用的环境变量  
PATH 决定了shell将到哪些目录中寻找命令或程序  
HOME 当前用户主目录  
HISTSIZE　历史记录数  
LOGNAME 当前用户的登录名  
HOSTNAME　指主机的名称  
SHELL 当前用户Shell类型  
LANGUGE 　语言相关的环境变量，多语言可以修改此环境变量  
MAIL　当前用户的邮件存放目录  
PS1　基本提示符，对于root用户是#，对于普通用户是$

注：公司每次用外部yum源，都需要允许login.py登录获取外网权限。

# 创建本地yum源：

## 创建本地yum源

1、安装createrepo包；  
2、新建本地源目录（假若本地源在当前家目录下）  
   2.1 $mkdir -p local-yum/pkgs/       #其中i386是包放置目录  
   2.2 将安装包放置到pkgs目录中  
   2.3 createrepo local-yum/pkgs           #创建一个源  
  
3、配置yum源（可以单独新建一个repo文件，也可以添加到已经存在的repo文件中）  
[local-yum]  
name=local-yum  
baseurl=file:///home/local-yum/pkgs（或http://xxx.xxx.xxx.xxx/pkgs）  
enabled=1  
gpgcheck=0

priority=1

不用createrepo会报错，缺少四个必要文件：filelists.xml.[gz]，other.xml.[gz]，primary.xml.[gz]和repomd.xml(md 意思是 metadata)，其中最主要的是repomd.xml文件。一般出现的找不到repodata目录或文件的错误的原因通常有三个：一是路径问题；二是没有生成repodate目录；三是\*.repo配置文件冲突（这个需要特意注意）。

createrepo --update:   
   有时候你的软件仓库中含有很多的软件包，虽然只是其中的一小部分有所改动但是你却不必为每个软件包重新产生元数据，这样做明显太浪费时间了。这就是--update 选项诞生的原因。  
   像之前一样运行 createrepo 命令但是添加了--update 选项.

       $createrepo --update local-yum/

   现在 createrepo 只更新那些在上次产生元数据之后被改变，添加或者删除了的项目.

verifytree +url

   如果你刚刚做了一个 repo 并且你想要验证在你的 repo 中的所有东西都是正确且可以完成其功能的话，你可以运行 verifytree 命令来检查所有的 repodata 和软件包。

如果报gpgcheck，可以加--nogpgcheck参数或者在/etc/yum.conf里改

# 取BMC日志、sel日志：

## 针对硬件问题

Telnet到BMC上，

telnet 128.1.1.槽位号:24 端口 ，user: zte ,password:zte

BMC上日志获取方法：

BMC上的日志为 /FLASH2/parsesel

在没有CMM的环境可以通过串口连接到BMC的串口，然后在[BMCMGR]下通过命令查看 :cat /FLASH2/parsesel

登录到BMC的web网站的方法

https://ip地址

user:zteroot, password: superuser

然后等到kvm路径查看

# repair filesyste模式修复方法：

## 对如下的问题修复

第一种情况：非正常关机引起的磁盘分区问题 不能正常进入系统  
如下问题一般是如何引起和应该怎么样解决呢？  
Finding module dependencies;  
modprobe:modprobe:can't locate module block-major-3  
fsck.ext3Possibly non-existent or swap device?  
such device or address while trying to open /dev/hdb1  
/boot: clena, 41/26104 files. 12679/104391 blocks[FAILED]  
\*\*\* AN error occurred during the file system check  
\*\*\* Dropping you to a shell; the systm will reboot  
\*\*\* when you leave the shell  
Give root password for maintenance  
(or type Control-D to continue):  
Login incorrect.  
(Repair filesystem) 1 #  
非正常关机引起的，用fsck /dev/hdaX后reboot进入，或只用fsck来修复，只管y回车(也可以使用fsck -y / )

第二种情况：由于/etc/fstab文件编辑错误 而引起的不能正常进入系统  
解决方法就是修改/etc/fstab文件成原来正常格式 删除错误的或者是不存在的挂载目录。

1、根据提示进入修复模式 Login incorrect.  
修复模式下（read-only system） 文件是被保护的不能修改，运行下面命令把系统文件权限改成可读写。  
（Repair filesystem) 1 # mount -o remount,rw /  
现在根目录就可以读写了．

2、修复/etc/fstab文件,  
(Repair filesystem) 1 # vim /etc/fstab  
保存 wq  
(Repair filesystem) 1 #reboot  
OK,正常界面又出来了。

# 下电流程分析：

## 查看日志

先分析我们的reboot.log日志，如没有

拷贝messages日志，搜索] start字段，是上电的开始

# 远程支持：

## kvm方式

ssh登录不了，有kvm的，可以从http浏览器访问ip。

进入后默认用户名/密码：zteroot/superuser

如果是服务器，会直接登录iSAC，可以看到基础信息，点kvm就能进入。

如果是机框管理，先选择要登录的刀片，再进iSAC，再点kvm进入。

（需要先配java，控制面板，java，安全，信任的站点）

## dashboard

远程控制主机ip，mstsc，登陆后用浏览器登录tecs，实例，控制台。

## 全球一张网，TeamViewer

提供ip和端口登录，远程控制。

# gerrit相关：

## 增加ftp本地用户的方法：

增加ftp本地用户的方法：  
    1、useradd tfger  
    2、passwd tfger  
    3、vi /etc/vsftpd/vsftpd.conf，保证有如下几行：  
       local\_enable=YES  
       chroot\_local\_user=NO  
       chroot\_list\_enable=YES  
       chroot\_list\_file=/etc/vsftpd/chroot\_list  
       user\_config\_dir=/etc/vsftpd/user\_conf  
    4、修改/etc/vsftpd/chroot\_list，增加一行tfger  
    5、增加/etc/vsftpd/user\_conf/tfger:  
       dirlist\_enable=YES  
       anon\_world\_readable\_only=NO  
       write\_enable=YES  
       anon\_upload\_enable=YES  
       anon\_mkdir\_write\_enable=YES  
       anon\_other\_write\_enable=YES  
       local\_root=/home/liushn/else  
       anon\_umask=022  
    6、重启vsftpd服务  
    注：如果主目录不需要另外指定，用默认的/home/tfger的话，就不需要另外设置/etc/vsftpd/user\_conf/tfger了

# 修改小系统镜像：

## initrd.img

initrd.img的解压和打包：

1)解包initrd.img命令: xz -cd initrd.img |cpio -idmv

或者：cpio -ivmd < initrd.img

2)修改解压的系统目录,可以将脚本放到指定的目录里

3)打包命令: find -path "./\*" -not -path '\*/.svn\*'| cpio -o -H newc | xz --format=lzma >/home/initrd.img

修改完小系统，修改syslinux.cfg里的配置选项，选择内核和小系统启动。

如果是gunzip压缩的：

cp /boot/initrd-\*\*\*.img initrd.img.gz  
gunzip initrd.img.gz  
mkdir initrd  
mv initrd.img initrd  
cd initrd  
cpio -ivmd < initrd.img

通过以上命令就将initrd.img解压了，现在就可以对其进行编辑，完成后使用以下命令重新压制

find . | cpio -o -H newc | gzip > /mnt/sda1/boot/initrd.img-2.6.18-4-686

## initramfs

initramfs的解压和打包：

/usr/lib/dracut/skipcpio /boot/initramfs-3.10.0-327.22.2.el7.x86\_64.img | zcat | cpio -imd

# 一些入口链接：

## 对内发布版本的链接

\\10.42.43.12\对内发布版本\计算存储平台\ZXTECS\TFGroup

## 加载selenium2Library失败，robotframework环境搭建

<http://www.bubuko.com/infodetail-555581.html>

## tfg系统根分区扩容方法

http://wxcop.zte.com.cn/topic/view/8888

## 空白盘启动虚机

http://opencos.zte.com.cn/opencos/read.php?tid=1268&fpage=2

见OS tfg安装升级方法

## pxe合入daisy步骤

http://wxcop.zte.com.cn/topic/view/17967

## 内存热插拔

https://www.kernel.org/doc/Documentation/memory-hotplug.txt

## Mimosa/TFG系统不同方式安装

http://dev.zte.com.cn/topic/view/14582