Linux 网络配置

Linux 网络配置

网卡绑定

将多个物理网卡绑定为一个逻辑网卡，从而加大IO速度并实现有效的网卡单一故障点的避免。

Linux 网络配置

网卡绑定模式

第一种模式 :mode=0 即 :(balance-rr) Round-robin policy（平衡轮询环策略）

特点：传输数据包顺序是依次传输（即：第 1 个包走 eth0 ，下一个包就走 eth1.... 一直循环下去，直到最后一个传输完毕），此模式提供负载平衡和容错能力；但是我们知道如果一个连接或者会话的数据包从不同的接口发出的话，中途再经过不同的链路，在客户端很有可能会出现数据包无序到达的问题，而无序到达的数据包需要重新要求被发送，这样网络的吞吐量就会下降

Linux 网络配置

网卡绑定模式

第二种模式：mode=2 ，即： (active-backup)

Active-backup policy（主 - 备份策略）

特点：只有一个设备处于活动状态，当一个宕掉另一个马上由备份转换为主设备。mac 地址是外部可见得，从外面看来，bond 的 MAC 地址是唯一的，以避免 switch(交换机) 发生混乱。此模式只提供了容错能力；由此可见此算法的优点是可以提供高网络连接的可用性，但是它的资源利用率较低，只有一个接口处于工作状态，在有 N 个网络接口的情况下，资源利用率为 1/N

Linux 网络配置

网卡绑定模式

第三种模式：mode=2 ，即：(balance-xor) XOR policy（平衡策略）

特点：基于指定的传输 HASH 策略传输数据包。缺省的策略是：(源 MAC 地址 XOR 目标 MAC 地址) % slave 数量。其他的传输策略可以通过 xmit_hash_policy 选项指定，此模式提供负载平衡和容错能力

Linux 网络配置

网卡绑定模式
第四种模式：mode=3，即：broadcast（广播策略）

特点：在每个slave接口上传输每个数据包，此模式提供了容错能力

Linux 网络配置

网卡绑定模式
第五种模式：mode=4 ，即：(802.3ad) IEEE 802.3adDynamic link aggregation（IEEE 802.3ad 动态链接聚合）

特点：创建一个聚合组，它们共享同样的速率和双工设定。根据 802.3ad 规范将多个 slave 工作在同一个激活的聚合体下。

Linux 网络配置

网卡绑定模式

mode=4

外出流量的 slave 选举是基于传输 hash 策略，该策略可以通过 xmit_hash_policy 选项从缺省的 XOR 策略改变到其他策略。需要注意的是，并不是所有的传输策略都是 802.3ad 适应的，尤其考虑到在 802.3ad 标准 43.2.4 章节提及的包乱序问题。不同的实现可能会有不同的适应性。

Linux 网络配置

网卡绑定模式
mode=4
必要条件：
条件 1 ： ethtool 支持获取每个 slave 的速率和双工设定

条件 2 ： switch(交换机) 支持 IEEE 802.3ad Dynamic link aggregation

条件 3 ：大多数 switch(交换机) 需要经过特定配置才能支持 802.3ad 模式

Linux 网络配置

命令 :ethtool

功能 : 查询或控制网络设备和硬件设置

语法结构 :ethtool [选项] [设备名]

示例 :

1. 查询 ethx 网口基本设置

   #ethtool enp0s3

Linux 网络配置

示例：

2. 显示 ethtool 命令帮助

   #ethtool –h

3. 查询 enp0s3 设备相关信息

   #ethtool –i enp0s3

4. 查询 enp0s3 设备注册信息

   #ethtool –d enp0s3

Linux 网络配置

示例：

5. 重置 enp0s3 网口到自适应模式

   #ethtool –r enp0s3

6. 查询 enp0s3 网口收发包统计

   #ethtool –S enp0s3

7. 设置网口速率 10/100/1000M 、设置网口半 / 全双工、设置网口是否自协商

   #ethtool –s enp0s3 [speed 10|100|1000] [duplex half|full] [autoneg on|off]

Linux 网络配置

网卡绑定模式

第六种模式：mode=5 ，即：(balance-tlb) Adaptive transmit load balancing（适配器传输负载均衡）

特点：不需要任何特别的 switch(交换机) 支持的通道 bonding 。在每个 slave 上根据当前的负载（根据速度计算）分配外出流量。如果正在接受数据的 slave 出故障了，另一个 slave 接管失败的 slave 的 MAC 地址。

该模式的必要条件：ethtool 支持获取每个 slave 的速率

Linux 网络配置

网卡绑定模式

第七种模式：mode=6，即：(balance-alb) Adaptive load balancing（适配器适应性负载均衡）

Linux 网络配置

网卡绑定模式

Mode=6

特点：该模式包含了 balance-tlb 模式，同时加上针对 IPV4 流量的接收负载均衡 (receive load balance, rlb)，而且不需要任何 switch(交换机) 的支持。接收负载均衡是通过 ARP 协商实现的。 bonding 驱动截获本机发送的 ARP 应答，并把源硬件地址改写为 bond 中某个 slave 的唯一硬件地址，从而使得不同的对端使用不同的硬件地址进行通信。

Linux 网络配置

网卡绑定

总结：

mode=0: 平衡轮询模式 balance-rr

mode=1: 主 - 备模式 active-backup

mode=2: 平衡策略 balance-xorg

mode=3: 广播策略 broadcast

mode=4:IEEE 802.3 动态链接聚合 802.3ad

mode=5: 适配器传输负载均衡 balance-tlb

mode=6: 适配器适应负载均衡 balance-alb

Linux 网络配置

网卡绑定

常用模式：

mode=0: 平衡轮询模式 balance-rr

mode=1: 主 - 备模式 active-backup

mode=2: 平衡策略 balance-xorg

mode=6: 适配器适应负载均衡 balance-alb

Linux 网络配置

网卡绑定的实现 (1)

1) 自动增加 bonding 模块，以下次启动时自动增加 bonding 模块

#vim /etc/modules-load.d/bonding.conf
#Load Bonding Modules
bonding

或手工加载模块

#modprobe bonding

Linux 网络配置

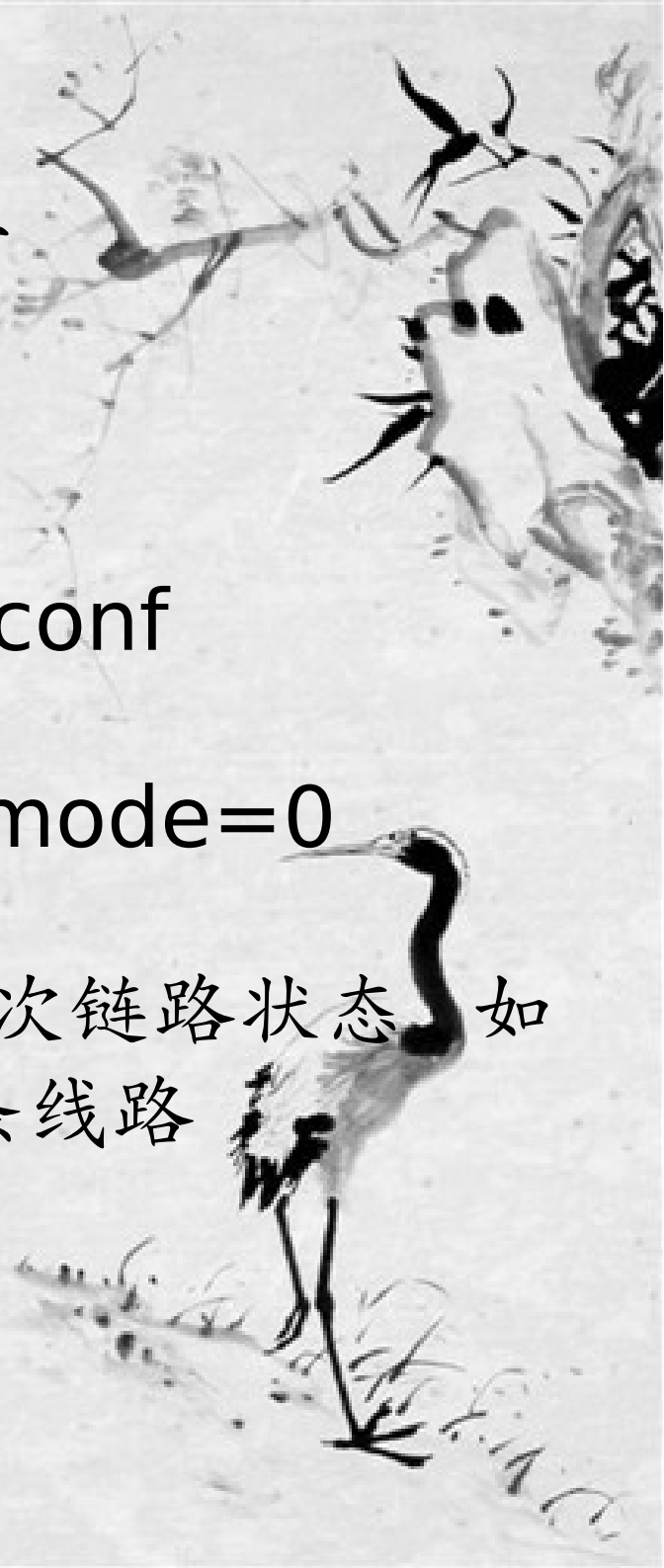
网卡绑定的实现(1)

2) 设置绑定的模式

```
#vim /etc/modprobe.d/bond.conf
alias bond0 bonding
options bond0 miimon=100 mode=0
```

//*miimon: 系统每多少毫秒检测一次链路状态，如有一条线路不通，则转换到另外一条线路

//*mode=0: 使用平衡轮询环策略



Linux 网络配置

网卡绑定的实现 (1)

3) 增加逻辑接口 br0

```
#brctl addbr br0
```

4) 将物理网卡加入至逻辑接口 br0

```
#brctl addif br0 enp0s8
#brctl addif br0 enp0s9
```

5) 配置 br0 的 IP 地址

```
#ifconfig br0 192.168.1.1
```

Linux 网络配置

网卡绑定的实现 (2)

1) 准备 bonding 模块自动加载

2) 定义 bond0, 模式采用 mode=0, 负载轮询

```
#nmcli con add type bond ifname bond0 mode balance-rr
```

3) 将网络设备加入至 bond0

```
#nmcli con add type bond-slave ifname enp0s8 master bond0
#nmcli con add type bond-slave ifname enp0s9 master bond0
```

Linux 网络配置

网卡绑定的实现 (2)

4) 确认网卡配置

```
#cd  /etc/sysconfig/network-scripts/
#ls -l bond-bond0
```

5) 默认 bond0 为 dhcp 客户端

```
#dhclient bond-bond0
```

Linux 网络配置

网卡绑定的实现 (2)

6) 设置 bond0 的 IP 地址

#nmcli con modify bond-bond0 ipv4.addresses “192.168.100.1/24 192.168.100.254”

7) 启用 bond0

#nmcli con up bond-bond0

8) 查看 bond0 的 IP

#ip addr sh bond0

Linux 网络配置

网卡绑定的实现 (2)

9) 断开指定设备

```
#nmcli dev dis bond-slave-en1
```

Linux 网络配置

网卡绑定的实现 (3)

1) 进入网络设备配置文件目录

#cd /etc/sysconfig/network-scripts

Linux 网络配置

网卡绑定的实现(3)

2) 编写 br0 文件(绑定)

```
#vim ifcfg-br0
DEVICE=br0
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=static
ONBOOT=yes
BONDING_OPTS=”mode=balance-rr miimon=10”
IPADDR=192.168.100.1
NETMASK=255.255.255.0
```

Linux 网络配置

网卡绑定的实现 (3)

2) 编写 enp0s8 文件 (绑定物理)

```
#vim ifcfg-br0
DEVICE=enp0s8
TYPE=Ethernet
NAME=enp0s8
ONBOOT=yes
MASTER=br0
SLAVE=yes
```

Linux 网络配置

网卡绑定的实现 (3)

3) 编写 enp0s9 文件 (绑定物理)

```
#vim ifcfg-br0
DEVICE=enp0s9
TYPE=Ethernet
NAME=enp0s9
ONBOOT=yes
MASTER=br0
SLAVE=yes
```

Linux 网络配置

网卡绑定的实现 (3)

4) 启动 network 服务

```
#systemctl restart network
```

5) 查看 br0 的 IP 地址

```
#ip addr sh br0
```

6) 查看 /proc 记录

```
#cat /proc/net/bonding/bond0
```

Linux 网络配置

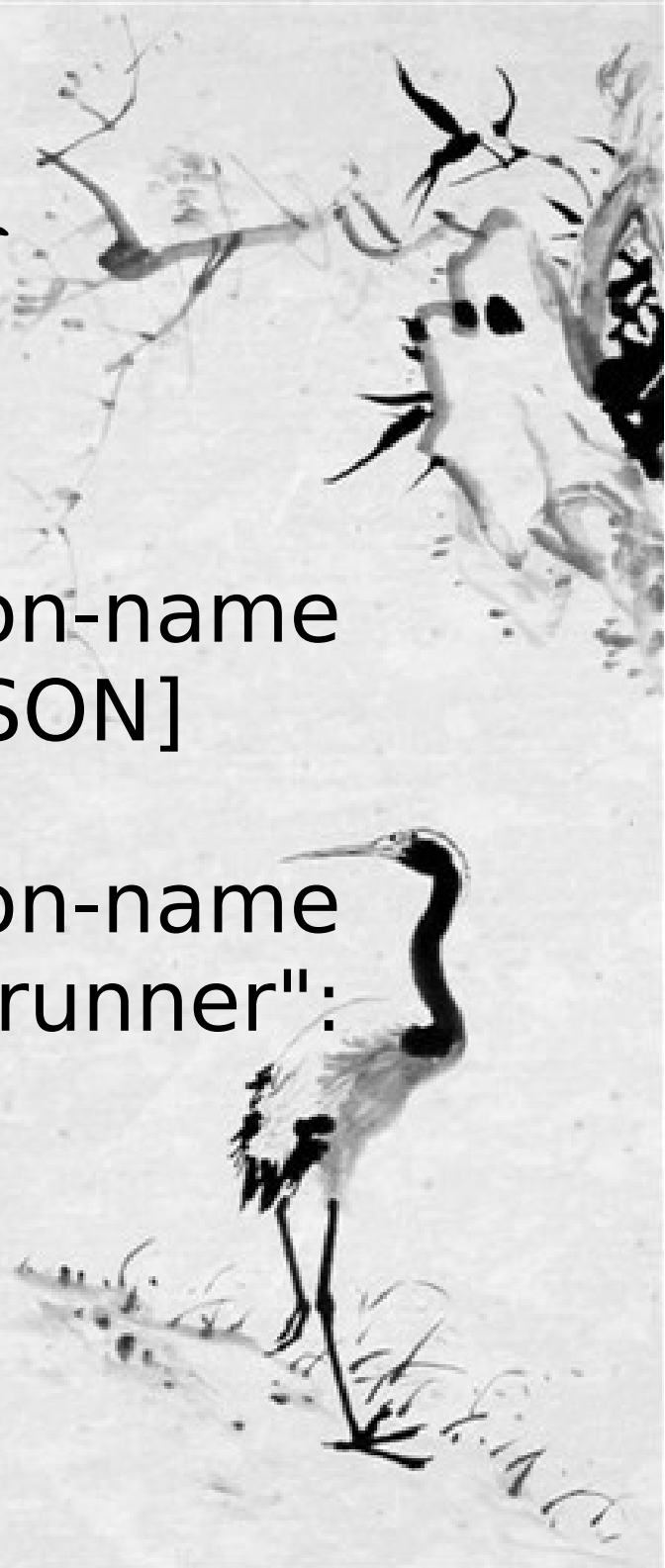
team 实现(1)

格式

#nmcli con add type team con-name CNAME ifname INAME [config JSON]

如

#nmcli con add type team con-name team0 ifname team0 config '{"runner": {"name": "loadbalance"}}'



Linux 网络配置

team 实现 (1)

#nmcli con mod team0 ipv4.addresses 1.2.3.4/24

nmcli con mod team0 ipv4.method manual

nmcli con add type team-slave con-name team-slave-enp0s3 ifname enp0s3 master team0

nmcli con add type team-slave con-name team-slave-enp0s8 ifname enp0s8 master team0

Linux 网络配置

team 实现 (1)

```
# nmcli con up team0
```

断开指定端口

```
#nmcli dev dis team0-port1
```

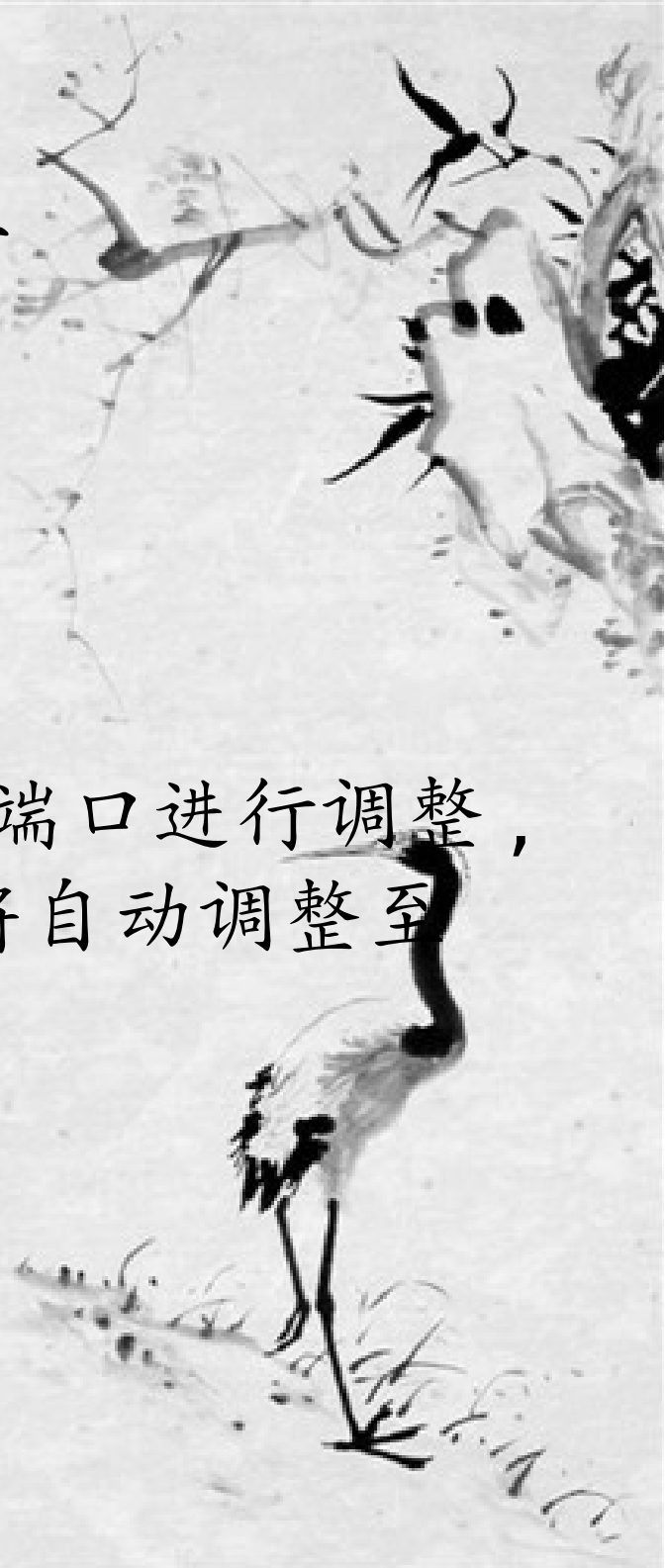
Linux 网络配置

team 管理

1. 查看 team0 中的端口

#teamnl team0 ports

//* 如果不存在，请使用 nmcli 将端口进行调整，可直接删除过去使用的端口。这样将自动调整至 team0



Linux 网络配置

team 管理

2. 查看当前活动的端口

#teamnl team0 getoption activeport

3. 设定活动端口

#teamnl team0 setoption activeport 3

4. 查看 team0 状态

#teamdctl team0 state

Linux 网络配置

team 管理

5. 查看当前 team0 配置

#teamdctl team0 config dump

Linux 网络配置

team 管理

6. 使用配置 (active-backup 模式)

cat /tmp/team.conf

```
{
    "device": "team0",
    "mcast_rejoin": {
        "count": 1
    },
    "notify_peers": {
        "count": 1
    },
```

Linux 网络配置

team 管理

6. 使用配置

```
"ports": {
    "eth1": {
    "prio": -10,
    "sticky": true,
        "link_watch": {
            "name": "ethtool"
        }
    },
```

Linux 网络配置

team 管理

6. 使用配置

//*

prio: 优先级

link_watch: 使用 ethtool 命令来监控接口的链路状态

*//

Linux 网络配置

team 管理

6. 使用配置

```
"eth2": {
    "prio": 100,
        "link_watch": {
            "name": "ethtool"
        }
    }
},
"runner": {
    "name": "activebackup"
```

Linux 网络配置

team 管理

6. 使用配置

}

}

nmcli con mod team0 team.config /tmp/team.conf

Linux 网络配置

team 实现 (2)

#cat /etc/sysconfig/network-scripts/ifcfg-team0

```
DEVICE=team0
DEVICETYPE=Team
TEAM_CONFIG="{\"runner\": {\"name\": \"broadcast\"}}"
BOOTPROTO=none
IPADDR0=1.2.3.4
PREFIX0=24
NAME=team0
ONBOOT=yes
```

Linux 网络配置

team 实现 (2)

```
# /etc/sysconfig/network-scripts/ifcfg-team0-enp0s3
DEVICE=enp0s3
DEVICETYPE=TeamPort
TEAM_MASTER=team0
NAME=team0-enp0s3
ONBOOT=yes
```

Linux 网络配置

team 实现 (2)

```
# /etc/sysconfig/network-scripts/ifcfg-team0-enp0s8
DEVICE=enp0s8
DEVICETYPE=TeamPort
TEAM_MASTER=team0
NAME=team0-enp0s8
ONBOOT=yes
```

Linux 网络配置

bridges 实现(1)

桥接：可实现虚拟网卡与物理网卡捆绑，从而实现虚拟网卡通信

#nmcli con add type bridge con-name br0 ifname br0

#nmcli con add type bridge-slave con-name br0-port1 ifname enp0s3 master br0

#nmcli con add type bridge-savle con-name br0-port2 ifname enp0s8 master br0

#ifconfig br0 1.2.3.4/24

#brctl shwo

Linux 网络配置

bridges 实现(1)

#cd /etc/sysconfig/network-scripts

#cat ifcfg-br0

#cat ifcfg-br0-port1

#cat ifcfg-br0-port2