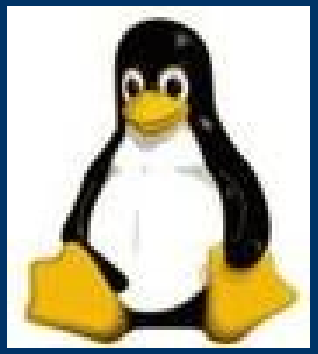
Linux 下的 Cluster 实现

啜立明



Linux 中
实现 HA 集群技术

Linux 下的 HA

- 双机技术

指由两台服务器运行某种同样的应用，为用户提供服务，当某一台出现问题时，用户的请求将由另一台服务器继续提供，从而实现高可用性。双机技术又被称为双机热备或双机容错

Linux 下的 HA

- 双机技术的实现不需要特定的硬件环境或者是操作系统 Kernel 的特定支持。因此仅需要双机 / 集群软件就可以实现

- 双机软件通过专用的信号传输通道，可以让两台服务器相互检测对方的状态，通过检测可得知对方如何。如对方出现问题可在第一时间作出反应

HA 容错运作过程

- Auto-Detect(自动检测)

通过两台主机所连接的线缆，经过负载的监听程序进行相互检测。其检测的内容有许多：

(1) 主机硬件
(2) 主机网络
(3) 主机操作系统
(4) 数据库引擎及其他应用程序
(5) 主机与磁盘整列连接线缆等

HA 容错运作过程

- Auto-Switch(自动切换)

　　如果某台主机确认对方出现故障，则将自动接手对方的工作来确保用户的请求可以得到及时处理

- Auto-Recover(自动恢复)

　　当故障主机修复完毕后可回归到生产系统中，通过一定配置可自动切换回以前状态继续工作

HA 的工作方式

- HA 的工作方式分为三种
- (1) 主从方式
- (2) 双机双工方式
- (3) 集群工作方式

HA 的工作方式

- 主从方式的工作原理

主机工作，备份机处于待命状态。当主机出现故障，备份机通过信号检测得知后将接管主机的一切工作，待主机回复正常后可以通过手工或自动配置切换到主机上运行。数据的一致性可通过其他技术解决

HA 的工作方式

- 双机双工方式

两台主机同时运行各自的服务工作且相互监督。当任何一台出现故障时，另一台会立即接管它的一切，保证工作的时效性。

HA 的工作方式

- 集群工作方式

多台主机一起工作，各自运行一个或多个服务，同时为每一个服务定义一个或多个备份主机。当主机出现故障时，备份主机将接管一切工作。

实现 Linux 下的 HA

- 实现 HA 的软件有许多种，其中包括

- 商业软件

(1)SteelEye 的 LifeKeeper for Linux
(2)Rose DataSystem 的 RoseHA
(3)Symantec 的 Verita

实现 Linux 下的 HA

- 实现 HA 的软件有许多种，其中包括

- 开源软件

(1)Heartbeat

(2)KeepAlived

实现 Linux 下的 HA

- Linux-HA 项目

Linux-HA 项目开创的目的就是提供一整套基于 Linux 的高可用性集群，其目标为 (RAS) 即：

Reliability(可靠性)

Availability(可用性)

Serviceablity(可服务性)

实现 Linux 下的 HA

- Heartbeat 本身属于 Linux-HA 项目的一个部分

- Hearbeat ：

  它通过在两台计算机之间建立一种类似于心跳一样的机制，当 Master 出现问题时 ,Slaver 可以通过像心跳一样的信号机制检测到故障并自动接管 Master 的一切工作。

  两台计算机可通过串行线缆或网络连接成为独立的点对点网络，这种线缆在 HA 中又称为”心跳”线

实现 Linux 下的 HA

- 获取 Heartbeat
- 在安装之前请确保系统已经安装了 Python 程序

- Debian/Ubuntu

#apt-get install heartbeat-2

- RedHat/CentOS

#yun install heartbeat

实现 Linux 下的 HA

- 配置 Heartbeat

- Heartbeat 的配置文件主要有三个

(1)ha.cf
(2)haresources
(3)authkeys

实现 Linux 下的 HA

- ha.cf

  Heartbeat 的主要配置文件，其控制 Heartbeat 的工作方式。如何时从 Master 切换到 Slaver, 并且根据什么状态切换回来

- haresources

  此配置文件控制当双方进行切换的时候，那种资源将被释放或应该被保留，哪些服务该被停止或者启动。

实现 Linux 下的 HA

- authkeys
  此配置文件是安全认证配置文件，可确保 Master 、 Slaver 双方的身份真实可靠

- ha.cf/haresources/authkeys 三个配置文件均保存在 /etc/ha.d 目录中

实现 Linux 下的 HA

**** 不完全解读 ha.cf****

- debugfile /var/log/ha-debug
  此为详细日志选项，一般用于调试使用

- logfile /var/log/ha-log
  logfile 是标准的、其他非调试信息的写入位置

实现 Linux 下的 HA

- logfacility local0

  写入到系统日志 (syslog) 的日志级别，默认为 local0

- keepalive 2

  主、备服务器每个多少时间检测一次通信，默认单位为秒

实现 Linux 下的 HA

- deadtime 30

  当经过多少秒后主、备服务器不能通信，则认为主服务器失效

实现 Linux 下的 HA

- warntime 10

  在日志中发出” late heartbeat” 警告之前等待多少秒

- initdead 120

  在计算机重启后，需要一段时间后网络才能进行正常工作。此选项专门用于处理系统重启时的情况。一般为 deadtime 时间的两倍

实现 Linux 下的 HA

- udpport 694
  用于测试使用的 udp 端口号

- baud 19200
  如果使用串口线缆作为“心跳”线缆，可设置波特率

- serial /dev/ttyS0
  指定串口线缆设备文件

实现 Linux 下的 HA

- bcast eth0

  如采用双绞线作为“心跳”线，应指定那块网卡作为“心跳”检测网卡

- mcast eth0 255.0.0.1 694 1 0

  设置多播地址及端口、 ttl 、 loopback 值

- unicast eth0 192.168.1.2

  如进行点对点进行“心跳”可使用 unicast

实现 Linux 下的 HA

- auto_fallback on

　　当主服务器从故障中恢复时，是否自动把服务器切换到主服务器 (on) ，还是继续由备份服务器继续服务直至手工干预 (off)

- node node_name

　　为 HA 集群加入节点，主服务器在最前面。

例 :

```
 node master
 node slaver
```

实现 Linux 下的 HA

**** 解读 haresources****

- 此文件主要控制整个的 HA 有哪些资源与服务，在出现或恢复故障时，应如何控制这些资源

- 此文件的配置格式为：

主机名 资源 1 资源 2 资源 3 ... 资源 n

- 例如

node1 192.168.10.1 httpd

实现 Linux 下的 HA

- node1 192.168.10.1 httpd

  表示 node1 是主服务器，其控制资源的 IP 地址为 192.168.10.1, 所控制的服务器为 httpd

实现 Linux 下的 HA

**** 解读 authkeys****

- auth 1

  表示以第一种方式进行验证

- 1 crc

  第一种方式验证的方法只是简单使用 CRC 进行效验，此种验证方法的特点是速度快、安全性低。仅适用于双机直连方式。

实现 Linux 下的 HA

**** 解读 authkeys****

- auth 2

  表示以第二种方式进行验证

- 2 sha1 HI!

  第二种方式验证的方法是使用 MD5 进行加密字符串” HI!” 后得到的密钥进行通信效验。 SHA1 的安全性最好，但对服务器性能要求很高。

实现 Linux 下的 HA

**** 解读 authkeys****

- auth 3

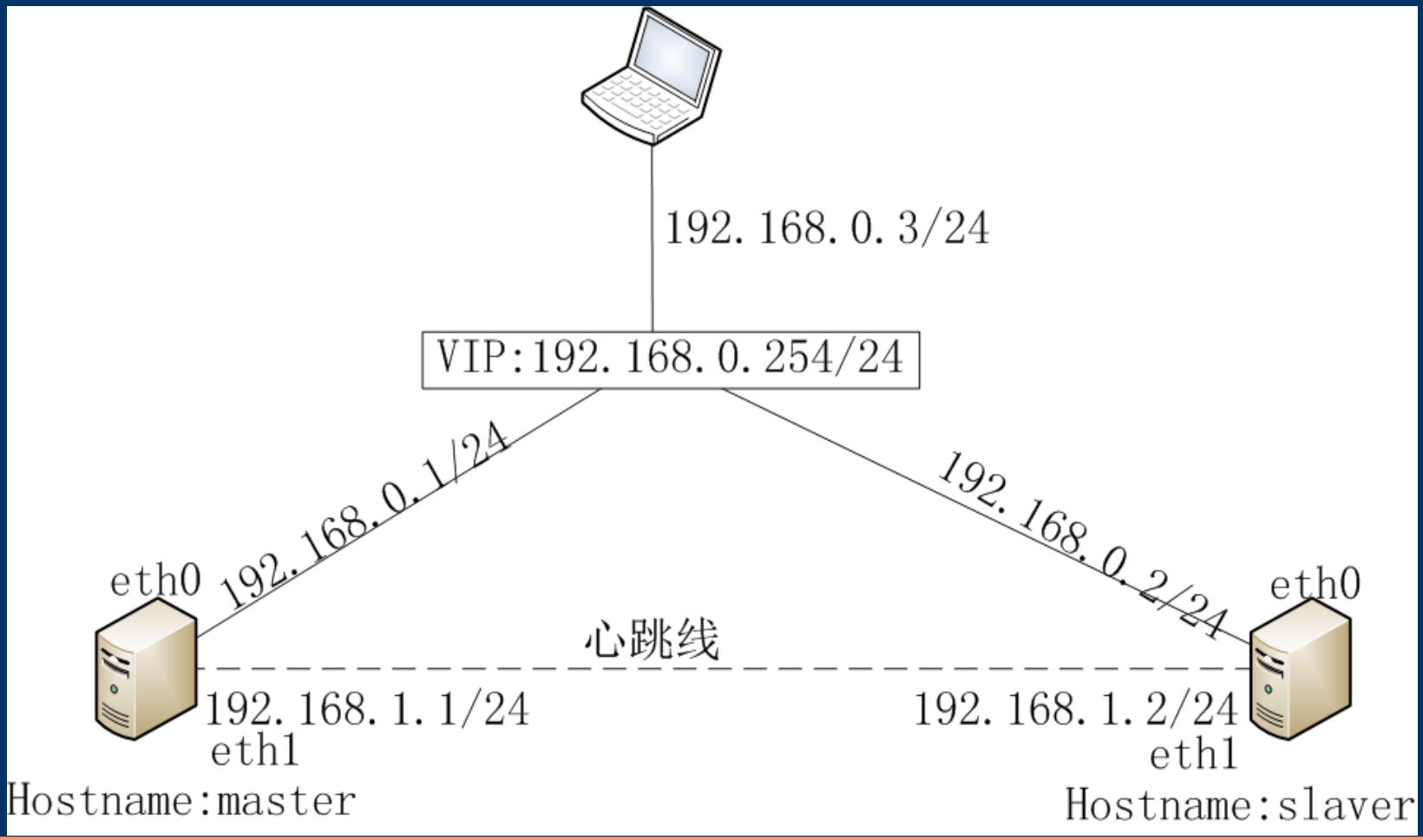
  表示以第二种方式进行验证

- 3 md5 Hello!

  第二种方式验证的方法是使用 MD5 进行加密字符串” Hello!” 后得到的密钥进行通信效验的。此种验证方式效率比 SHA1 高，安全性比 CRC 高。

实现 Linux 下的 HA

- (1) 试验拓扑图



实现 Linux 下的 HA

- (1) 为两台计算机设置固定 IP 地址及主机名，主机名必须要用 uname -n 可以看到。

- (2) 在两台计算机中 /etc/hosts 中加入以下信息：

  192.168.0.1 master
  192.168.1.1 master
  192.168.0.2 slaver
  192.168.1.2 slaver

实现 Linux 下的 HA

- （ 3)ping 测试网络连通性

- (4) 将 /usr/share/doc/heartbeat-2.1.x/ 目录下的 authkeys 、 haresources 、 ha.cf 三个文件复制到 /etc/ha.d 目录中。

- (5) 进入 /etc/ha.d 目录准备配置 HA

实现 Linux 下的 HA

- (6) 编辑 authkeys 文件，设定双机验证方式

```
#vim authkeys
// 在文件尾部追加
auth 1
1 crc
```

实现 Linux 下的 HA

- (7) 编辑 haresources 文件，设定主服务器及虚拟 IP 地址与所控制的资源

#vim haresources
// 在文件尾部追加以下内容
master 192.168.0.254 httpd

实现 Linux 下的 HA

- (8) 编辑 ha.cf, 配置 HA 基本特性

#vim ha.cf
// 打开以下注释字段内容

logfile /var/log/ha-log

logfacility local0

keepalive 2

实现 Linux 下的 HA

- (8) 编辑 ha.cf, 配置 HA 基本特性（续）

deadtim 30

warntim 10

initdead 120

实现 Linux 下的 HA

- (8) 编辑 ha.cf, 配置 HA 基本特性（续）

udpport 694

bcast eth1

实现 Linux 下的 HA

- (8) 编辑 ha.cf, 配置 HA 基本特性 (续)

auto_failback on

// 找到” node kathy” 字段，在其下行增加 HA 集群节点，主节点在第一行
Node master
Node slaver

// 保存退出

实现 Linux 下的 HA

- (9) 配置 slaver 服务器，过程参照 2-8 步骤

- (10) 启动 HA
#service heartbeat start

- (11) 查看 master 服务器中 eth0:0 是否有 IP ，其 IP 地址是否为 192.168.0.254, 并查看进程确定 httpd 已经启动

实现 Linux 下的 HA

- (12) 配置客户端
- (13) 创建两台 Apache 的索引文件，每个索引文件内容不同
- (14) 客户端进行 WEB 浏览
- (15) 停止 master 服务器
- (16) 客户端浏览 WEB 是否可以看到 slaver 的主页，如可以代表 HA 成功

实现 Linux 下的 HA

- (17) 重启 master 服务器
- (18) 客户端重新浏览 WEB, 是否能够重新看到 master 服务器上的主页内容，如可以则代表 HA 的自动切换能力实现

HA 集群实现

- 使用 Heartbeat 实现 Linux 下的 HA
- 试验目的 : 掌握 Heartbeat 实现 HA 集群
- 试验人员 : 个人
- 所需要计算机设备 : 至少 3 台计算机
- 试验时间 :30 分钟

Linux 下的 Cluster 实现

结　　束

master.chuai@gmail.com

304630723

152990419