**Linux 高可用（HA）集群之Corosync 詳解**

大綱  
一**、**Corosync概述  
二**、**Corysync與Pacemaker安裝  
三、Corosync詳細配置  
四、Corosync啟動信息

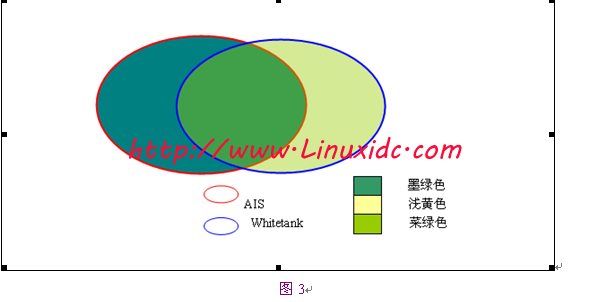
一**、**Corosync概述      
1.簡單概述  
  要說明corosync的由來，首先我們得從AIS說起，然後說明OpenAIS，最後才會說到corosync**。**  
2.AIS概述  
  應用介面規範（AIS）是用來定義應用程式介面（API）的開放性規範的集合，這些應用程式作為中間件為應用服務提供一種開放**、**高移植性的程式介面。是在實現高可用應用過程中是亟需的**。**服務可用性論壇（SA Forum）是一個開放性論壇，它開發並發布這些免費規範。使用AIS規範的應用程式介面（API），可以減少應用程式的複雜性和縮短應用程式的開發時間，這些規範的主要目的就是為了提高中間組件可攜性和應用程式的高可用性**。**SAF AIS是一個開放性工程，在不斷更新中。  
3.OpenAIS概述  
  OpenAIS是基於SA Forum標準的集群框架的應用程式介面規範**。**OpenAIS提供一種集群模式，這個模式包括集群框架，集群成員管理，通信方式，集群監測等，能夠為集群軟件或工具提供滿足AIS標準的集群介面，但是它沒有集群資源管理功能，不能獨立形成一個集群**。**OpenAIS組件包括AMF,CLM,CKPT,EVT,LCK,MSG，TMR,CPG,EVS等，因OpenAIS分支不同，組件略有不同**。**（下面介紹）OpenAIS主要包含三個分支：Picacho，Whitetank，Wilson。Wilson是最新的，比較穩定的版本是從openais 1**.**0.0到openais1**.**1.4。Whitetank現在是主流分支版本，比較穩定的版本是openais0.80到openais0**.**86。Picacho第一代的OpenAIS的分支，比較穩定的版本是openais0.70和openais0.71**。**現在比較常用的是Whitetank和Wilson，兩者之間有很多不同。OpenAIS從Whitetank升級到Wilson版本後，組件變化很大，Wilson把Openais核心架構組件獨立出來放在Corosync（Corosync是一個集群管理引擎）裡面**。**Whitetank包含的組件有AMF，CLM，CKPT，EVT，LCK ,MSG, CPG，CFG,EVS, aisparser, VSF\_ykd，bojdb等**。**而Wilson只含有AMF，CLM，CKPT，LCK, MSG,EVT,TMR（TMR，Whitetank裡面沒有），這些都是AIS組件**。**其他核心組件被放到了Corosync內。Wilson被當做Corosync的一個外掛程式**。**(詳細請查看官方文檔)   
4.Corosync概述  
  Corosync是OpenAIS發展到Wilson版本後衍生出來的開放性集群引擎工程**。**可以說Corosync是OpenAIS工程的一部分**。**OpenAIS從openais0.90開始獨立成兩部分，一個是Corosync；另一個是AIS標準介面Wilson**。**Corosync包含OpenAIS的核心框架用來對Wilson的標準介面的使用、管理**。**它為商用的或開源性的集群提供集群執行框架。Corosync執行高可用應用程式的通信組系統，它有以下特徵：  
一個封閉的程式組（A closed process group communication model）通信模式，這個模式提供一種虛擬的同步方式來保證能夠複製服務器的狀態**。**

一個簡單可用性管理組件（A simple availability manager），這個管理組件可以重新啟動應用程式的進程當它失敗後**。**

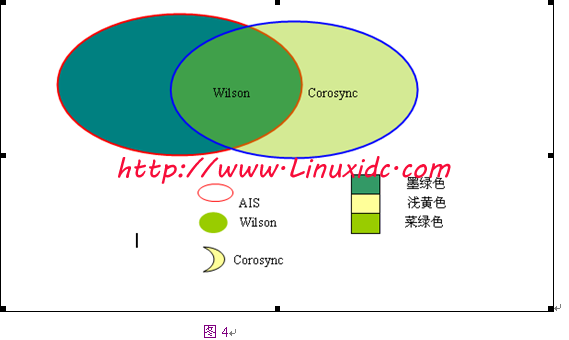
一個配置和內存數據的統計（A configuration and statistics in-memory database），內存數據能夠被設置，回復，接受通知的更改資訊**。**

一個定額的系統（A quorum system?）,定額完成或者丟失時通知應用程式**。**

5.AIS、OpenAIS,Corosync的關係  
(1).AIS與Whitetank的關係  
由圖3，可以看出，OpenAIS的分支版本Whitetank除了包含AIS標準的應用程式介面，同時也有自己獨立的管理模塊，這些獨立的模塊為圖3中淺黃色部分，包含CPG，CFG,EVS, aisparser, VSF\_ykd，bojdb等控制模塊**。**



(2).AIS與Wilson的關係  
當OpenAIS到了Wilson以後，OpenAIS一分為二，Wilson的組件基本都是AIS組件**。**其他控制的核心組件被添加到Corosync中，關係如圖4所示。



(3).Corosync與OpenAIS關係

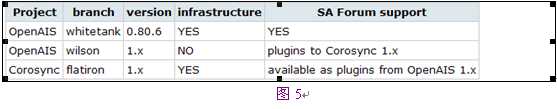
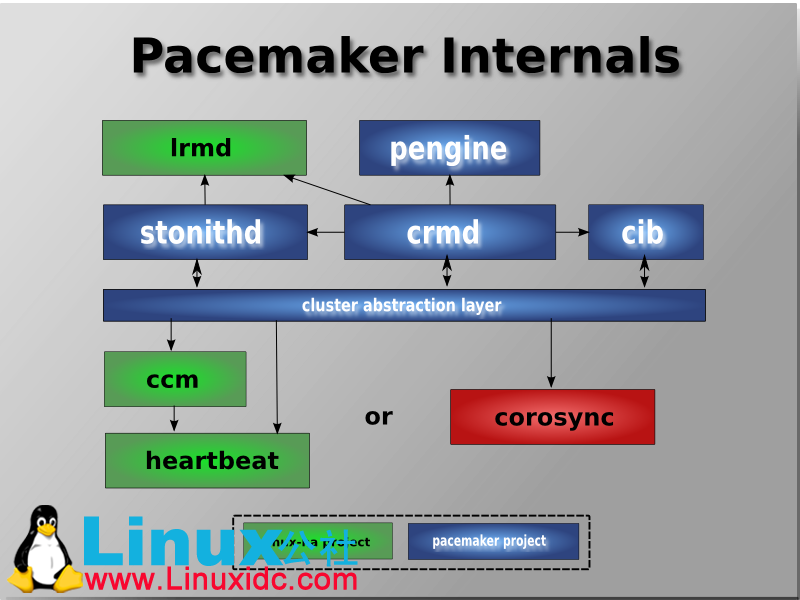


圖5所示，Wilson與Whitetank的主要區別在於Wilson相比Whitetank缺少核心架構**。**Wilson 作為Corosync的外掛程式支持SA Forum 。

6.OpenAIS集群實例  
(1).CMAN   
  CMAN是紅帽RHCS套件的核心部分，CCS是CMAN集群配置系統，配置cluster.conf，而cluster.conf其實就是openais的設定檔，通過CCS映射到openais**。**  
(2).Pacemaker1.x+corosync1.x   
  Pacemaker升級到1**.**0版本後，從Heartbeat獨立出來，Pacemaker achieves maximum availability for your cluster services by detecting and recovering from node and service-level failures**.** It achieves this by utilizing the messaging and membership capabilities provided by your preferred cluster infrastructure (currently either Corosync or Heartbeat)   
7**.**總結  
  簡單的說，AIS就是一個通用的應用程式編程介面，OpenAIS是AIS的子項目，標準的集群框架的應用程式介面規範，而corosync是OpenAIS是具體實現**。**這樣說大家應該很好理解吧！  
8.corosync版本  
官方網站：[http://corosync.github**.**io/corosync/](http://corosync.github.io/corosync/)，目前官方同在維護兩個版本分別為，  
corosync 1**.**x –> corosync 1**.**4.6（最新）代號：flatiron

corosync 2.x –> corosync 2.3**.**1（最新） 代號：needle

9.corosync與pacemaker組合



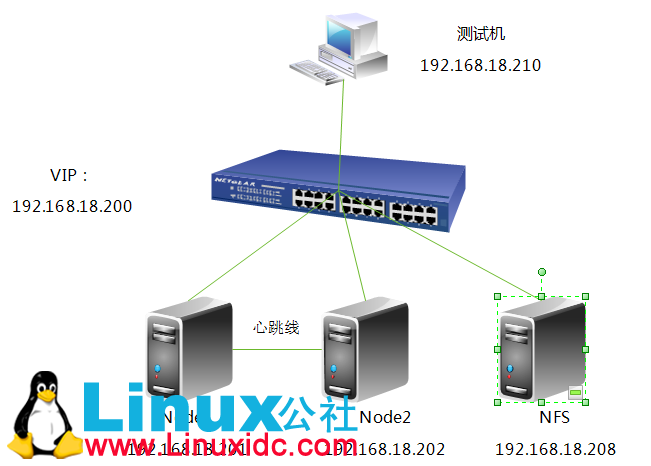
  從上圖中我們可以看到，不管heartbeat，還是corosync都是高可用集群中的Cluster Messaging Layer（集群資訊層），是主要傳遞發集群資訊與心跳資訊的，並沒有資源管理功能，資源管理還得依賴於上層的crm(Cluster resource Manager，集群資源管理器)，最著名的資源管理器，就是pacemaker，它是heartbeat v3分離出去的子項目**。**而現在corosync+pacemaker成了高可用集群中的最佳組合**。**好了，到這裡corosync與pacemaker的基礎知識就說到這裡了，下面我們來看看怎麼安裝corosync與pacemaker**。**

二**、**Corysync與Pacemaker 安裝  
1.環境說明  
(1).作業系統  
[CentOS](http://www.linuxidc.com/topicnews.aspx?tid=14)6.4 X86\_64位元系統

(2)**.**軟體環境  
corosync-1.4.1-15.el6\_4**.**1.x86\_64

pacemaker-1.1.8-7**.**el6.x86\_64

(3).拓撲環境



2**.**前提條件  
node1:  
(1).各節點之間主機名稱互相解析  
[root@node1 ~]# uname -n    
node1.test.com   
[root@node1 ~]# vim /etc/hosts   
127**.**0.0.1  localhost localhost.localdomain localhost4 localhost4.localdomain4    
::1        localhost localhost.localdomain localhost6 localhost6.localdomain6      
192**.**168.18.201  node1.test.com  node1      
192.168**.**18.202  node2.test.com  node2   
[root@node1 ~]# ping node1    
[root@node1 ~]# ping node2

(2)**.**各節點之間時間同步

1 [root@node1 ~]# ntpdate 210**.**72.145.44

(3)**.**各節點之間ssh互信

[root@node1 ~]# ssh-keygen  -t rsa -f ~/.ssh/id\_rsa  -P ''      
[root@node1 ~]# ssh-copy-id -i .ssh/id\_rsa.pub root@node2.test.com

node2:  
(1)**.**各節點之間主機名稱互相解析  
[root@node2 ~]# uname -n    
node2.test.com   
[root@node2 ~]# vim /etc/hosts   
127**.**0.0.1  localhost localhost.localdomain localhost4 localhost4.localdomain4    
::1        localhost localhost.localdomain localhost6 localhost6.localdomain6      
192**.**168.18.201  node1.test.com  node1      
192**.**168.18.202  node2.test.com  node2   
[root@node2~]# ping node1    
[root@node2 ~]# ping node2

(2)**.**各節點之間時間同步

[root@node2 ~]# ntpdate 210**.**72.145**.**44

(3).各節點之間ssh互信

[root@node2 ~]# ssh-keygen  -t rsa -f ~/.ssh/id\_rsa  -P ''      
[root@node2 ~]# ssh-copy-id -i .ssh/id\_rsa.pub root@node1.test.com

3**.**配置yum源(EPEL源)  
node1:  
[root@node1 src]# wget http://download**.**[Fedora](http://www.linuxidc.com/topicnews.aspx?tid=5)project.org/pub/epel/5/x86\_64/epel-release-5-4.noarch.rpm   
[root@node1 src]# rpm -ivh epel-release-5-4.noarch.rpm   
warning: epel-release-5-4.noarch.rpm: Header V3 DSA signature: NOKEY, key ID 217521f6   
Preparing..**.**                ########################################### [100%]   
1:epel-release          ########################################### [100%]   
[root@node1 src]# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5   
[root@node1 src]# yum list

node2:

[root@node2 src]# wget http://download.fedoraproject.org/pub/epel/5/x86\_64/epel-release-5-4.noarch.rpm   
[root@node2 src]# rpm -ivh epel-release-5-4.noarch.rpm   
warning: epel-release-5-4.noarch.rpm: Header V3 DSA signature: NOKEY, key ID 217521f6   
Preparing..**.**                ########################################### [100%]   
1:epel-release          ########################################### [100%]   
[root@node2 src]# rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-5   
[root@node2 src]# yum list

4**.**關閉防火牆與SELinux

node1:  
[root@node1 ~]# service iptables stop   
[root@node1 ~]# vim /etc/selinux/config   
# This file controls the state of SELinux on the system**.**   
# SELINUX= can take one of these three values:   
#      enforcing - SELinux security policy is enforced**.**   
#      permissive - SELinux prints warnings instead of enforcing**.**   
#      disabled - SELinux is fully disabled**.**   
SELINUX=disabled   
# SELINUXTYPE= type of policy in use**.** Possible values are:   
#      targeted - Only targeted network daemons are protected**.**   
#      strict - Full SELinux protection.   
SELINUXTYPE=targeted

node2:  
[root@node2 ~]# service iptables stop   
[root@node2 ~]# vim /etc/selinux/config   
# This file controls the state of SELinux on the system**.**   
# SELINUX= can take one of these three values:   
#      enforcing - SELinux security policy is enforced**.**   
#      permissive - SELinux prints warnings instead of enforcing**.**   
#      disabled - SELinux is fully disabled**.**   
SELINUX=disabled   
# SELINUXTYPE= type of policy in use**.** Possible values are:   
#      targeted - Only targeted network daemons are protected**.**   
#      strict - Full SELinux protection**.**   
SELINUXTYPE=targeted

5.安裝corosync與pacemaker  
node1:  
12 [root@node1 ~]# yum install -y corosync\*   
[root@node1 ~]# yum install -y pacemaker\*

node2:  
[root@node2 ~]# yum install -y corosync\*   
[root@node2 ~]# yum install -y pacemaker\*

三**、**Corosync 詳細配置  
1.提供設定檔

[root@node1 ~]# cd /etc/corosync/    
[root@node1 corosync]# ll      
總用量 24      
-rw-r--r-- 1 root root  445 5月  15 05:09 corosync.conf.example      
-rw-r--r-- 1 root root 1084 5月  15 05:09 corosync.conf.example.udpu      
drwxr-xr-x 2 root root 4096 5月  15 05:09 service**.**d      
drwxr-xr-x 2 root root 4096 5月  15 05:09 uidgid**.**d

注：大家可以看到提供一個樣例檔corosync.conf.example

[root@node1 corosync]# cp corosync.conf.example corosync.conf

2**.**修改設定檔  
[root@node1 corosync]# cat corosync.conf    
# Please read the corosync.conf**.**5 manual page      
compatibility: whitetank   
totem {    
    version: 2      
    secauth: on #啟動認證      
    threads: 2      
    interface {      
        ringnumber: 0      
        bindnetaddr: 192**.**168.18**.**0 #修改心跳線網段      
        mcastaddr: 226.99.10.1 #組播傳播心跳資訊      
        mcastport: 5405      
        ttl: 1      
    }      
}   
logging {    
    fileline: off      
    to\_stderr: no      
    to\_logfile: yes    
    to\_syslog: no      
    logfile: /var/log/cluster/corosync.log #日誌位置      
    debug: off      
    timestamp: on      
    logger\_subsys {      
        subsys: AMF      
        debug: off      
    }      
}   
amf {    
    mode: disabled      
}   
#啟用pacemaker   
service {    
    ver: 0      
    name: pacemaker      
}   
aisexec {    
    user: root      
    group: root      
}

注：用 man corosync.conf 可以查看所有選項的意思**。**  
3.生成金鑰檔  
注：corosync生成key檔會預設調用/dev/random亂數設備，一旦系統中斷的IRQS的亂數不夠用，將會產生大量的等待時間，因此，為了節約時間，我們在生成key之前講random替換成urandom，以便節約時間**。**  
[root@node1 corosync]# mv /dev/{random,random.bak}      
[root@node1 corosync]# ln -s /dev/urandom /dev/random   
[root@node1 corosync]# corosync-keygen      
Corosync Cluster Engine Authentication key generator**.**      
Gathering 1024 bits for key from /dev/random**.**      
Press keys on your keyboard to generate entropy**.**      
Writing corosync key to /etc/corosync/authkey**.**

4.查看生成的key文件  
[root@node1 corosync]# ll    
總用量 24      
-r-------- 1 root root  128 8月  13 14:16 authkey      
-rw-r--r-- 1 root root  521 8月  13 11:11 corosync.conf      
-rw-r--r-- 1 root root  445 5月  15 05:09 corosync.conf.example      
-rw-r--r-- 1 root root 1084 5月  15 05:09 corosync.conf.example.udpu      
drwxr-xr-x 2 root root 4096 5月  15 05:09 service**.**d      
drwxr-xr-x 2 root root 4096 5月  15 05:09 uidgid**.**d

5**.**將key文件authkey與設定檔corosync.conf複製到node2上  
[root@node1 corosync]# scp -p authkey corosync.conf node2:/etc/corosync/    
authkey                                                                                100%  128    0**.**1KB/s  00:00      
corosync.conf                                                                          100%  521    0**.**5KB/s  00:00    
[root@node2 ~]# cd /etc/corosync/    
[root@node2 corosync]# ll      
總用量 24      
-r-------- 1 root root  128 8月  13 14:16 authkey      
-rw-r--r-- 1 root root  521 8月  13 11:11 corosync.conf      
-rw-r--r-- 1 root root  445 5月  15 05:09 corosync.conf.example      
-rw-r--r-- 1 root root 1084 5月  15 05:09 corosync.conf.example.udpu      
drwxr-xr-x 2 root root 4096 5月  15 05:09 service**.**d      
drwxr-xr-x 2 root root 4096 5月  15 05:09 uidgid**.**d

注：corosync 到這裡配置全部完成。下面我們進行啟動測試！  
四**、**Corosync 啟動資訊  
1.啟動corosync

[root@node1 ~]# ssh node2 "service corosync start"    
Starting Corosync Cluster Engine (corosync): [確定]      
[root@node1 ~]# service corosync start      
Starting Corosync Cluster Engine (corosync):              [確定]

2**.**查看corosync啟動資訊

(1).查看corosync引擎是否正常啟動  
[root@node1 ~]# grep -e "Corosync Cluster Engine" -e "configuration file" /var/log/cluster/corosync.log      
Aug 13 14:20:15 corosync [MAIN  ] Corosync Cluster Engine ('1**.**4.1'): started and ready to provide service**.**      
Aug 13 14:20:15 corosync [MAIN  ] Successfully read main configuration file '/etc/corosync/corosync.conf'**.**      
Aug 13 17:08:51 corosync [MAIN  ] Corosync Cluster Engine ('1**.**4.1'): started and ready to provide service.      
Aug 13 17:08:51 corosync [MAIN  ] Successfully read main configuration file '/etc/corosync/corosync.conf'**.**      
Aug 13 17:08:51 corosync [MAIN  ] Corosync Cluster Engine exiting with status 18 at main**.**c:1794.

(2)**.**查看初始化成員節點通知是否正常發出

[root@node1 ~]# grep  TOTEM /var/log/cluster/corosync.log      
Aug 13 14:20:15 corosync [TOTEM ] Initializing transport (UDP/IP Multicast)**.**      
Aug 13 14:20:15 corosync [TOTEM ] Initializing transmit/receive security: libtomcrypt SOBER128/SHA1HMAC (mode 0)**.**      
Aug 13 14:20:15 corosync [TOTEM ] The network interface [192**.**168**.**18.201] is now up.      
Aug 13 14:20:15 corosync [TOTEM ] A processor joined or left the membership and a new membership was formed**.**      
Aug 13 14:20:40 corosync [TOTEM ] A processor joined or left the membership and a new membership was formed**.**

(3).檢查啟動過程中是否有錯誤產生

[root@node1 ~]# grep ERROR: /var/log/cluster/corosync.log      
Aug 13 14:20:15 corosync [pcmk  ] ERROR: process\_ais\_conf: You have configured a cluster using the Pacemaker plugin for Corosync**.** The plugin is not supported in this environment and will be removed very soon**.**      
Aug 13 14:20:15 corosync [pcmk  ] ERROR: process\_ais\_conf:  Please see Chapter 8 of 'Clusters from Scratch' (http://www.clusterlabs.org/doc) for details on using Pacemaker with CMAN

(4)**.**查看pacemaker是否正常啟動  
[root@node1 ~]# grep pcmk\_startup /var/log/cluster/corosync.log      
Aug 13 14:20:15 corosync [pcmk  ] info: pcmk\_startup: CRM: Initialized      
Aug 13 14:20:15 corosync [pcmk  ] Logging: Initialized pcmk\_startup      
Aug 13 14:20:15 corosync [pcmk  ] info: pcmk\_startup: Maximum core file size is: 18446744073709551615      
Aug 13 14:20:15 corosync [pcmk  ] info: pcmk\_startup: Service: 9      
Aug 13 14:20:15 corosync [pcmk  ] info: pcmk\_startup: Local hostname: node1.test.com

3**.**查看集群狀態  
[root@node1 ~]# crm\_mon   
Last updated: Tue Aug 13 17:41:31 2013    
Last change: Tue Aug 13 14:20:40 2013 via crmd on node1.test.com      
Stack: classic openais (with plugin)      
Current DC: node2.test.com - partition with quorum      
Version: 1**.**1.8-7.el6-394e906      
2 Nodes configured, 2 expected votes      
0 Resources configured**.**   
Online: [ node1.test.com node2.test.com ]

注：大家可以看到，集群運行正常，node1與node2都線上，DC是node2節點**。**但是還沒有配置資源，配置資源就要用到pacemaker，在下一篇博客中我們重點講解pacemaker使用**。**用pacemaker來增加各種資源。^\_^……