Centro de Procesamiento de Datos
Proyecto Servicio Alta Disponibilidad

Servicio de consulta de información bursátil con recursos de alta disponibilidad.

Objetivo:

Se trata de desarrollar un sistema redundante de almacenamiento de datos bursátiles basado en Python.

Desarrollo:

- Primero instalar Eclipse desde: http://www.eclipse.org/downloads/
- Añadir el plugin de Python para Eclipse: http://pydev.org/manual_101_install.html
- Crear un repositorio en GitHub: https://github.com/

Como en la práctica anterior, se van a utilizar los nodos compute-0-0 y compute-0-1 para crear un configuración redundante pero basada en un modelo master/slave. Veamos cómo funciona:

El nodo que actúa como master consulta de forma periódica al servicio de Yahoo de datos bursátiles y envía al nodo slave la información recibida para que esté duplicada. Ambos nodos almacenan en una BBDD de Sqlite dichos datos. Cuando el nodo máster cae, el nodo slave debe pasar a modo máster para seguir con la consulta. Los datos recibidos no copiados al nodo remoto se marcan como no sincronizados. Cuando el nodo vuelve a estar activo se envían los datos no sincronizados al otro nodo.

Esta práctica está basada en Python. Algunas referencias a este lenguaje:

- http://www.python.org/doc/essays/ppt/lwnyc2002/intro22.ppt
- http://www.tutorialspoint.com/python/python_quick_guide.htm (Guía rápida)
- http://rgruet.free.fr/PQR27/PQR2.7_printing_a4.pdf (Guía rápida)

Para elaborar la práctica son necesarios algunos elementos que se describen a continuación:

Cómo recibir los datos bursátiles de Yahoo. Podemos partir del siguiente código:

```
#!/usr/bin/env python

import urllib2

def getYahooStockQuote(symbol):
    url = "http://download.finance.yahoo.com/d/quotes.csv?s=
%s&f=sl1d1c1hgv" % symbol
    f = urllib2.urlopen(url)
    s = f.read()
    f.close()
    s = s.strip()
    L = s.split(',')
    D = {}
    D['symbol'] = L[0].replace('"','')
    D['last'] = L[1]
    D['date'] = L[2]
    D['change'] = L[3]
    D['high'] = L[4]
    D['low'] = L[5]
    D['vol'] = L[6]
    return D

print (getYahooStockQuote('GOOG'))
```

Para comunicar dos programas en Python utilizaremos Pyro4. http://pythonhosted.org/Pyro4/
Acceso a la BBDD sqlite desde Python:

Consultar http://docs.python.org/2/library/sqlite3.html

También es posible mediante SQLAlchemy:

Consultar: http://www.pythoncentral.io/introductory-tutorial-python-sqlalchemy/

Desarrollar dos programas en Python (preferentemente en versión 3): un servidor y un cliente.
El programa servidor debe acceder a Yahoo para obtener los datos en tiempo real (cada 15 minutos) y almacenarlos en una BBDD local en sqlite. Este servidor se conecta a otra máquina (modo pasivo) para actualizar su BBDD remota. Por otra parte, el segundo servidor comprueba si está activa la primera máquina y cuando detecta que no lo está pasa al modo activo realizando las consultas a Yahoo. Las dos máquinas deben sincronizar sus datos cuando funcionen correctamente.
Por otra parte, el cliente debe poder acceder a uno u otro servidor para consultar los datos.

Extension opcional:
Ampliar los conocimiento sobre la configuración de recursos de alta disponibilidad (Pacemaker + Corosync). Se trata de utilizar el modo master/slave en un Resource Agent con Pacemaker, creando un agente (Resource Agent) master/slave para gestionar una aplicación Python.

Vamos a crear tres configuraciones:

- Tenemos un programa en Python que esté ejecutándose y muestre si está en estado máster o slave.
- En la segunda configuración, el nodo cuando se reactiva manda una orden de "sincronización" antes de pasar a master.
- En la tercera configuración tenemos 1 máster y 2 slaves. De forma que sólo uno de ellos es máster. Cuando un máster pierda la sincronizción, debe solicitar la sincronización de otro nodo que esté activo.

Hay que crear un Agente de Recurso (Resource Agent) nuevo. Estos agentes son programas o scripts que están instalados en /usr/lib/ocf. Crear una carpeta cpd y en ella crear el script partiendo de /usr/lib/ocf/resource.d/heartbeat/Stateful

Se puede consultar la información de Agentes de recurso en :

- http://doc.opensuse.org/products/draft/SLE-HA/SLE-ha-guide_sd_draft/cha.ha.agents.html

Información útil sobre Corosync + Pacemaker:

- http://doc.opensuse.org/products/draft/SLE-HA/SLE-ha-guide_sd_draft/cha.ha.manual_config.html