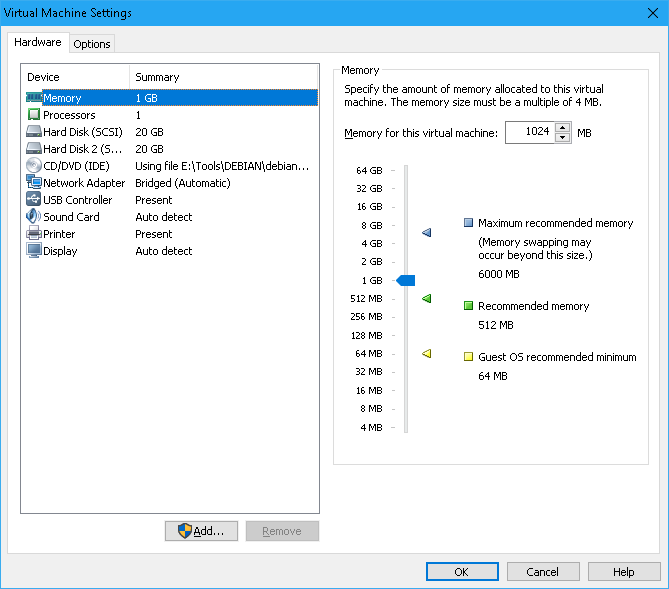
**Migración de Servers con DMTCP en el DVS**

Se utiliza VMWARE Workstation 15.

# Creación de NODE0

El primer nodo será NODE0. Crear una VM y utilizar como disco virtual el suministrado en partes.



La configuración del Network Adapter está en Bridge porque yo lo tengo conectado a mi modem ADSL. Esto me permite salir a internet desde la VM para instalar actualizaciones y buscar nuevo software y utilizar el GIT.

Además, la propia PC host, también pertenece a la misma LAN por lo cual puedo utilizar SSH tanto para disponer de una terminal SSH como de transferencia de archivos entre la PC host y cualquiera de las VMs.

La dirección IP del NODE0 es 192.168.0.100 y para NODE1 debe ser 192.168.0.101 y asi sucesivamente. La dirección IP se encuentra configurada en:

root@node0:/usr/src/dvs/dvs-apps# cat /etc/network/interfaces

# The loopback network interface

auto lo

iface lo inet loopback

# The primary network interface

auto eth0

iface eth0 inet static

address 192.168.0.100

netmask 255.255.255.0

network 192.168.0.0

broadcast 192.168.0.255

IMPORTANTE. No borrar la interface de LOOPBACK!!

En el archivo /etc/hosts están registrados todos los nodos del cluster de hasta 32 nodos.

192.168.0.100 node0

192.168.0.101 node1

192.168.0.102 node2

192.168.0.103 node3

192.168.0.104 node4

……..

Esto permite usar el nombre del nodo en lugar de su IP.

En /etc/hostname esta registrado el nombre del nodo.

root@node0:~# cat /etc/hostname

node0

En /etc/spread.conf están registrados todos los nodos que conforman el cluster spread.

root@node0:~# cat /etc/spread.conf | grep node

node0

node1

node2

node3

node4

…….

Pruebe:

* hacer ping desde su PC al NODO0 y viceversa usando direcciones IP.
* hacer SSH desde su PC al NODO0 mediante la dirección IP.
* Para transferencias de archivos se puede utilizar desde Windows WinSCP.
* Probar la salida a internet y la resolución de nombres.

root@node0:/usr/src/dvs/dvs-apps# ping www.linux.org

PING www.linux.org (104.27.166.219) 56(84) bytes of data.

64 bytes from 104.27.166.219: icmp\_seq=1 ttl=57 time=211 ms

--- www.linux.org ping statistics ---

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 211.928/211.928/211.928/0.000 ms

root@node0:/usr/src/dvs/dvs-apps#

Esta es la versión de kernel que utilizaremos.

root@node0:~# uname -a

Linux node0 4.9.88-DVK #157 SMP Wed Sep 16 22:44:57 -03 2020 i686 GNU/Linux

Y esta la version de Debian

root@node0:~# cat /etc/debian\_version

9.12

Que se denomina “***stretch***” (https://www.debian.org/releases/index.es.html)

# CREACION DEL NODOx A PARTIR DEL NODE0

* Crear otro directorio con el nombre del nodo correspondiente.
* Apagar la VM de NODE0, para evitar conflictos.
* Copiarle los archivos de disco (.vmdk) de la VM de NODE0 al directorio de NODEx
* Crear la VM de NODEx apuntando al directorio correspondiente.
* Encender la VM de NODEx
* Modificar los archivos
  + /etc/network/interfaces
  + /etc/hostname
* Rebootear y comprobar que el nombre y la dirección IP son correctas.

root@node0:/usr/src/dvs/dvs-apps# **hostname**

node0

root@node0:/usr/src/dvs/dvs-apps# **ifconfig**

**eth0**: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

**inet 192.168.0.100 netmask 255.255.255.0 broadcast 192.168.0.255**

inet6 fe80::20c:29ff:fe9a:7aa4 prefixlen 64 scopeid 0x20<link>

ether 00:0c:29:9a:7a:a4 txqueuelen 1000 (Ethernet)

RX packets 675 bytes 70779 (69.1 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 410 bytes 46237 (45.1 KiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

device interrupt 19 base 0x2000

**lo:** flags=73<UP,LOOPBACK,RUNNING> mtu 65536

**inet 127.0.0.1 netmask 255.0.0.0**

inet6 ::1 prefixlen 128 scopeid 0x10<host>

loop txqueuelen 1 (Local Loopback)

RX packets 0 bytes 0 (0.0 B)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 0 bytes 0 (0.0 B)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Luego arrancar la VM de NODE0 probar:

* Ping desde NODE0 a NODEx y viceversa.
* Ping desde NODEx al Host y viceversa.
* SSH y SCP desde el Host a NODEx
* Probar la salida a internet y la resolución de nombres.

root@node0:/usr/src/dvs/dvs-apps# ping www.linux.org

PING www.linux.org (104.27.166.219) 56(84) bytes of data.

64 bytes from 104.27.166.219: icmp\_seq=1 ttl=57 time=211 ms

--- www.linux.org ping statistics ---

1 packets transmitted, 1 received, 0% packet loss, time 0ms

rtt min/avg/max/mdev = 211.928/211.928/211.928/0.000 ms

root@node0:/usr/src/dvs/dvs-apps#

# LOCALIZACION DE LOS ARCHIVOS DE SUB-PROYECTOS

Y el código Fuente está organizado de la siguiente forma en **/usr/src**

dvs

├── chardev-ioctl << example of a char device and IOCTL usage

├── dvk-lib << library of dvk\_calls

├── dvk-loops << series of microbenchmarks for local IPC

├── dvk-mod << the dvk kernel module

├── dvs-apps << Native DVS applications

│   ├── dvsd << Gestion distribuida del DVS cluster

│   ├── radar << Seguimiento y localizacion de servicios

│   └── dvs\_run << arranque de procesos sin bind - migración

├── dvk-proxies << proxies

│   └── test << microbenchmarks for remote IPC

├── dvk-tests << dvk\_calls test programs

├── vos << dvk\_calls test programs

│   ├── uk << unikernel VOS source code

│   └── mol << MoL source code

├── include

│   ├── com << common header files (for kernel, lib, users)

│   ├── dvk << header only used by the dvk

│   └── **dvs** << to be used other dvs header files

├── lkm\_example << example of kernel module

├── lz4-1.8.2 << LZ4 to be used by proxies, VOS or apps.

├── reljmp-master << to get unpublished kernel functions and vars

├── **linux-4.9.88 << Linux kernel modified source programs**

│   └── ipc << DVK integration to SYSV IPC

│   └── dvk-mod << the dvk kernel module

└── spread-src-5.0.1 << the spread toolkit

En el directorio **include/com** están las definiciones de constantes, estructuras de datos, códigos de error, etc. que son COMUNES a todas las aplicaciones de DVS.

Estos archivos no se deberían modificar dado que impactarían en todas las aplicaciones.

Ahora hay un directorio denominado ***vos*** específico para los Virtual Operating Systems (VOS).

Actualmente existen 2 subdirectorios: ***mol***, para Minix Over Linux y ***uk*** para un unikernel basado el Lightweight IP (LwIP - este no fue migrado aún a esta versión de DVS)

Para los siguientes proyectos deberían crearse nuevos directorios bajo ***vos***/, como por ejemplo:

* Directorio ***rk***: para Rumpkernel
* Directorio ***uml***: para User Mode Linux
* Directorio ***vrouter*** : para Contrail dpdk Vrouter

En tanto que para los proyectos de aplicaciones del DVS utilizaremos el directorio ***dvs-apps***:

* Directorio ***radar***: para RADAR
* Directorio ***webmin***: para lo requerido por el módulo webmin de gestión del DVS.
* Directorio ***dvsd***: para el demonio de monitoreo general del DVS.
* Directorio ***dvs\_run***: Permite arrancar procesos haciendo binding externo. Además están los comandos para migración de procesos.

# ARRANQUE DEL DVS EN NODE0 y NODE1

Se va a arrancar el DVS en el NODE0 utilizando un script que puede ser distinto según el proyecto. Por ahora se arrancará usando un script genérico.

Se arranca el DVS, se inicia un DC (Distributed Container), se arranca un proxy TCP para comunicar con el NODEx.

El script se encuentra en /usr/src/dvs/dvk-tests, y se usa de la siguiente forma:

usage: ./tests.sh <lcl\_nodeid> <dcid>

***lcl\_nodeid***: es el ID del nodo local, si estamos en NODE0 el valor será 0.

***dcid*** :El ID del DC a registrar en este nodo.

La siguiente es la salida por pantalla de ejecutar tests.sh en el NODE0 y registrar el DC0 en el. Durante su ejecución realizará varias detenciones para poder ir viendo el resultado de sus acciones. Solo dar ENTER.

root@node0:/usr/src/dvs/dvk-tests# ./tests.sh 0 0

lcl\_nodeid=0 dcid=0

Enter para continuar...

Spread Enter para continuar...

dvk 344064 0

mount Enter para continuar...

partition 5

mount Enter para continuar...

local\_nodeid=0 Enter para continuar...

DEBUG 862:dvk\_open:108: Open dvk device file /dev/dvk

Initializing DVS. Local node ID 0...

DEBUG 862:dvk\_dvs\_init:559: nodeid=0

DEBUG 862:dvk\_dvs\_init:571: ioctl ret=0 errno=0

DEBUG 862:dvk\_dvs\_init:576: ioctl ret=0

d\_name=TEST\_CLUSTER d\_nr\_dcs=32 d\_nr\_nodes=32 d\_nr\_procs=221 d\_nr\_tasks=35 d\_nr\_sysprocs=64

d\_max\_copybuf=65536 d\_max\_copylen=1048576

d\_dbglvl=FFFFFF version=5 flags=0 sizeof(proc)=0

Get DVS info

DEBUG 862:dvk\_getdvsinfo:265:

DEBUG 862:dvk\_getdvsinfo:274: ioctl ret=0 errno=0

DEBUG 862:dvk\_getdvsinfo:279: ioctl ret=0

local node ID 0...

d\_name=TEST\_CLUSTER d\_nr\_dcs=32 d\_nr\_nodes=32 d\_nr\_procs=221 d\_nr\_tasks=35 d\_nr\_sysprocs=64

d\_max\_copybuf=65536 d\_max\_copylen=1048576

d\_dbglvl=FFFFFF version=5 flags=101 sizeof(proc)=512

DC0 Enter para continuar...

TCP LZ4 BAT PROXY Enter para continuar...

Active Internet connections (servers and established)

Proto Recv-Q Send-Q Local Address Foreign Address State

tcp 0 0 0.0.0.0:3001 0.0.0.0:\* LISTEN

tcp 0 0 0.0.0.0:4001 0.0.0.0:\* LISTEN

tcp 0 0 0.0.0.0:16385 0.0.0.0:\* LISTEN

tcp 0 0 0.0.0.0:4803 0.0.0.0:\* LISTEN

tcp 0 0 0.0.0.0:5001 0.0.0.0:\* LISTEN

tcp 0 0 0.0.0.0:80 0.0.0.0:\* LISTEN

tcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN

tcp 0 100 192.168.0.100:22 192.168.0.196:55358 ESTABLISHED

tcp6 0 0 :::22 :::\* LISTEN

Enter para continuar... ID Flags Proxies -pxsent- -pxrcvd- 10987654321098765432109876543210 Name

0 6 -1 0 0 -------------------------------X node0

1 2 1 0 0 -------------------------------- node1

Proxies Flags Sender Receiver --Proxies\_Name- 10987654321098765432109876543210

1 1 868 869 node1 ------------------------------X-

ID Type -lpid- -flag- -misc- -pxsent- -pxrcvd- -getf- -sendt -wmig- name

1 send 868 0 1 0 0 27342 27342 27342 lz4tcp\_proxy\_ba

1 recv 869 0 1 0 0 27342 27342 27342 lz4tcp\_proxy\_ba

ADDNODE Enter para continuar...

dcid=0

flags=0

nr\_procs=221

nr\_tasks=34

nr\_sysprocs=64

nr\_nodes=32

dc\_nodes=1

dc\_pid=864

warn2proc=27342

warnmsg=-1

dc\_name=DC0

nodes 33222222222211111111110000000000

10987654321098765432109876543210

-------------------------------X

cpumask=1

DEBUG 878:dvk\_open:108: Open dvk device file /dev/dvk

Adding node 1 to DC 0...

DEBUG 878:dvk\_add\_node:505: dcid=0 nodeid=1

DEBUG 878:dvk\_add\_node:517: ioctl ret=0 errno=0

DEBUG 878:dvk\_add\_node:522: ioctl ret=0

ID Flags Proxies -pxsent- -pxrcvd- 10987654321098765432109876543210 Name

0 6 -1 0 0 -------------------------------X node0

1 2 1 0 0 -------------------------------X node1

dcid=0

flags=0

nr\_procs=221

nr\_tasks=34

nr\_sysprocs=64

nr\_nodes=32

dc\_nodes=3

dc\_pid=864

warn2proc=27342

warnmsg=-1

dc\_name=DC0

nodes 33222222222211111111110000000000

10987654321098765432109876543210

------------------------------XX

cpumask=1

DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name

run # . /dev/shm/DC0.sh

Luego ejecutar

root@node0:/usr/src/dvs/dvk-tests# . /dev/shm/DC0.sh

En el NODE1 arrancar el script con los siguientes argumentos

root@node1:/usr/src/dvs/dvk-tests# ./tests.sh 1 0

Salida similar…

root@node1:/usr/src/dvs/dvk-tests# . /dev/shm/DC0.sh

Para verificar el estado de la comunicación entre ambos.

root@**node0**:/usr/src/dvs/dvk-tests# cat /proc/dvs/nodes

ID Flags Proxies -pxsent- -pxrcvd- 10987654321098765432109876543210 Name

0 6 -1 0 0 -------------------------------X node0

1 2 1 0 0 -------------------------------X node1

root@**node1**:/usr/src/dvs/dvk-tests# cat /proc/dvs/nodes

ID Flags Proxies -pxsent- -pxrcvd- 10987654321098765432109876543210 Name

0 E 0 0 0 -------------------------------X node0

1 6 -1 0 0 -------------------------------X node1

Esto nos dice que el node0 y node1 contienen proceso del DC0 (columna con X)

Los siguientes archivos nos dan información respecto a los proxies y los nodos que comunican.

root@**node0**:/usr/src/dvs/dvk-tests# cat /proc/dvs/proxies/info

Proxies Flags Sender Receiver --Proxies\_Name- 10987654321098765432109876543210

1 1 808 809 **node1** ------------------------------**X**-

root@**node1**:/usr/src/dvs/dvk-tests# cat /proc/dvs/proxies/info

Proxies Flags Sender Receiver --Proxies\_Name- 10987654321098765432109876543210

0 7 651 650 **node0** -------------------------------**X**

# INICIO DEL DC0

El **DC0** está configurado con los siguientes parámetros que se encuentran definidos en el archivo /dev/shm/**DC0.cfg** el cual es creado “*al vuelo*” por el **script tests.sh** que ya ejecutamos.

root@node0:/usr/src/dvs/dvk-tests# cat /dev/shm/DC0.cfg

# dc\_init config file

dc DC0 {

dcid 0;

nr\_procs 221;

nr\_tasks 34;

nr\_sysprocs 64;

nr\_nodes 32;

tracker "NO";

memory 512;

image "/usr/src/dvs/vos/images/debian0.img";

mount "/usr/src/dvs/vos/rootfs/DC0";

};

El script tests.sh a través del programa **dc\_init** crea el archivo **DC0.sh** el cual contiene instrucciones para setear y exportar variables de entorno como el **NODEID** y el **DC0**.

El valor almacenado en la variable de entorno **DC0** es el del PID del proceso **dc\_init** (demonio) que queda ejecutando mientras el **DC0** viva. Es fundamental conservar ese valor para que otros procesos puedan compartir el mismo Contenedor utilizando el comando ***nsenter***.

root@node0:/usr/src/dvs/dvk-tests# more /dev/shm/DC0.sh

#!/bin/bash

NODEID=0

DC0=864

export NODEID

export DC0

**EN NODE0**

root@node0:/usr/src/dvs/dvs-apps/dc\_init# echo $DC0

**536**

root@node0:/usr/src/dvs/dvs-apps/dc\_init# ps -ef | grep dc\_init

root **536** 1 0 08:43 ? 00:00:00 /usr/src/dvs/dvs-apps/dc\_init/dc\_init /dev/shm/DC0.cfg

root 897 494 0 08:48 pts/0 00:00:00 grep dc\_init

**NOTA: Todo NAMESPACE en Linux debe tener al menos un proceso registrado. En el DC0 el PID del proceso registrado está en la variable de entorno $DC0 que en este caso vale 536. El comando *nsenter* permite ejecutar un proceso dentro de un NAMESPACE. La opción *–p* indica que se crea un nuevo namespace para los PIDs y la opción –t indica que se utilice el namespace del proceso indicado, en este caso 536.**

**EN NODE 1**

root@node1:/usr/src/dvs/dvk-proxies/test# echo $DC0

**503**

root@node1:/usr/src/dvs/dvk-proxies/test# ps -ef | grep dc\_init

root **503** 1 0 10:00 ? 00:00:00 /usr/src/dvs/dvs-apps/dc\_init/dc\_init /dev/shm/DC0.cfg

root 528 480 0 10:06 pts/0 00:00:00 grep dc\_init

# MIGRACION DE PROCESOS: ESCENARIO 1

**Escenario PREVIO**

**migr\_server: NODE0**

**migr\_client: NODE1**

**Escenario FINAL:**

**migr\_server: NODE1**

**migr\_client: NODE1**

A los efectos de mostrar el funcionamiento se utilizarán múltiples terminales en cada uno de los Nodos.

|  |  |
| --- | --- |
| **NODE0** | **NODE1** |
| * /usr/src/lkl/dmtcp/bin/dmtcp\_coordinator -p 1234 –-daemon -q |  |
| * cd /usr/src/dvs/dvk-tests/ * ./tests.sh 0 0 * . /dev/shm/DC0.sh * nsenter -p -t$DC0 ./test\_rmtbind 0 11 1 migr\_client * cd /usr/src/dvs/dvs-apps/dvs\_run * rm \*.dmtcp | * cd /usr/src/dvs/dvk-tests/ * ./tests.sh 1 0 * . /dev/shm/DC0.sh * nsenter -p -t$DC0 ./test\_rmtbind 0 10 0 migr\_server * cd /usr/src/dvs/dvs-apps/dvs\_run * rm \*.dmtcp |
| root@node0:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 11 11 -1/-1 1 1000 0 27342 27342 27342 27342 migr\_client | root@node1:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 -1/-1 0 1000 0 27342 27342 27342 27342 migr\_server |
| * /usr/src/lkl/dmtcp/bin/dmtcp\_launch -h node0 -p 1234 ./migr\_server 0 10 & |  |
|  | * nsenter -p -t$DC0 ./migr\_client 0 11 10 4096 100 1 & |
|  | * ./dvs\_migrate -s 0 10 |
| * ./dvs\_migrate -s 0 10 |  |
| root@node0:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 939/939 0 800 20 27342 27342 27342 27342 migr\_server  0 11 11 -1/-1 1 1000 0 27342 27342 27342 27342 migr\_client | root@node1:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 -1/-1 0 1800 0 27342 27342 27342 27342 migr\_server  0 11 11 867/3 1 8000 20 27342 27342 10 27342 migr\_client |
| * /usr/src/lkl/dmtcp/bin/dmtcp\_command -h node0 -p 1234 -bc * mv ckpt\_\*.dmtcp checkpoint\_migr\_server.dmtcp |  |
| * sshpass -p "root" scp -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -r checkpoint\_migr\_server.dmtcp root@node1:/usr/src/dvs/dvs-apps/dvs\_run |  |
|  | * dmtcp\_restart checkpoint\_migr\_server.dmtcp & |
| root@node0:~# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 11 11 -1/-1 1 1000 0 27342 27342 27342 27342 migr\_client | root@node1:~# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 889/-1 1 8 0 31438 27342 27342 27342 DMTCP:migr\_serv  0 11 11 867/3 1 0 20 27342 27342 27342 27342 migr\_client |
|  | * ./dvs\_migrate -s 0 10 |
|  | root@node1:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 889/-1 1 800 0 27342 27342 27342 27342 DMTCP:migr\_serv  0 11 11 867/3 1 8000 20 27342 27342 10 27342 migr\_client |
|  | * ./dvs\_migrate -r 0 10 |
|  | root@node1:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 889/-1 1 8 0 31438 27342 27342 27342 DMTCP:migr\_serv  0 11 11 867/3 1 0 20 27342 27342 27342 27342 migr\_client |

1. En NODE0 se arranca el coordinador de DMTCP porque allí se encuentra el servidor. Este coordinador se podría arrancar en todos los nodos del DVS.
2. En NODE0 se inicial el DVS, se configura y se inicia el DC0 en donde van a ejecutar los procesos de pruebas. Se arrancan los proxies contra el NODE1 y se adiciona el NODE1 al DC0. Esto quiere decir que desde el NODE0 se pueden referencia procesos del DC0 que se encuentren en el NODE1. Luego se setea la variable de entorno $DC0 que contiene el PID del proceso asociado al DC0. También se establece que el proceso ***migr\_client*** es un proceso REMOTO (flag=1000) y se encuentra en el NODE1 (nd=1). Luego, se borran del directorio de trabajo todos los archivos de checkpoint del DMTCP.
3. En NODE1, se inicializan el DVS, el DC0, se arrancan los proxies contra NODE0 y se establece que en NODE0 pueden ejecutar procesos del DC0. También se establece que el proceso ***migr\_server*** es un proceso REMOTO (flag=1000) y se encuentra en el NODE0 (nd=0). Luego, se borran del directorio de trabajo todos los archivos de checkpoint del DMTCP.
4. En NODE0, se arranca a través del lanzador de DMTCP la ejecución del ***migr\_server***.
5. En NODE1, se arranca utilizando ***nsenter*** la ejecución del ***migr\_client***. En este momento comienzan a transferir datos entre ambos procesos localizados en diferentes nodos.
6. En NODE1, se le notifica al DVK que el proceso ***migr\_server*** (dcid=0, endpoint=10) va a iniciar (start) un proceso de migración (flag=1800). En este momento se detienen las comunicaciones y el ***migr\_client*** queda en espera de la finalización de la migración (flag=8000). **ATENCION: Este comando de migración SIEMPRE debe ejecutarse primero en los nodos en donde se encuentran los procesos Client, de lo contrario, al no responder el Server, el Client puede dar por finalizada la comunicación**.
7. En NODE0, se le notifica al DVK que el proceso ***migr\_server*** (dcid=0, endpoint=10) va a iniciar (start) un proceso de migración.
8. En NODE0, se le solicita al coordinador de DMTCP realizar un Checkpoint bloqueante (bc) lo que provoca que se genere el archivo de checkpoint. Luego, se normaliza el nombre del archivo de checkpoint.
9. En NODE0, se transfiere el archivo de Checkpoint al NODO1 (nodo destino) utilizando ***ssh***. Para automatizar la transferencia se utiliza ***sshpass*** de tal forma de que no solicite la password del usuario de NODE1.
10. En NODE1, se rearranca el proceso ***migr\_server*** a partir de la imagen del DMTCP. El ***migr\_server*** está preparado para la migración que le solicita al DVK que lo considere migrado en el NODO1 (*dvk\_migr\_commit()*). Esta operación reinicia automáticamente las comunicaciones de todos los procesos que se estaban comunicando con el ***migr\_server*** y que fueron detenidos al realizar *dvk\_migr\_start().* En este momento se puede ver que las comunicaciones continúan entre Client y Server.
11. En NODE1. Al solo efecto de probar que ocurriría ante una migración fallida, se notifica al DVK que se inicia una migración del proceso de Endpoint=10. En este momento se vuelven a detener las comunicaciones.
12. En NODE1, se simula una migración fallida y que el proceso Server continuará ejecutando en su nodo origen (ahora NODE1). Para ello se le comunica el DVK que realice un Rollback (*dvk\_migr\_rollback()*) y reinicie las comunicaciones.

# MIGRACION DE PROCESOS: ESCENARIO 2

**Escenario PREVIO**

**migr\_server: NODE1**

**migr\_client: NODE1**

**Escenario FINAL:**

**migr\_server: NODE0**

**migr\_client: NODE1**

A los efectos de mostrar el funcionamiento se utilizarán múltiples terminales en cada uno de los Nodos.

|  |  |
| --- | --- |
| **NODE0** | **NODE1** |
|  | * /usr/src/lkl/dmtcp/bin/dmtcp\_coordinator -p 1234 –-daemon -q |
| * cd /usr/src/dvs/dvk-tests/ * ./tests.sh 0 0 * . /dev/shm/DC0.sh * nsenter -p -t$DC0 ./test\_rmtbind 0 11 1 migr\_client * nsenter -p -t$DC0 ./test\_rmtbind 0 10 1 migr\_server * cd /usr/src/dvs/dvs-apps/dvs\_run * rm \*.dmtcp | * cd /usr/src/dvs/dvk-tests/ * ./tests.sh 1 0 * . /dev/shm/DC0.sh * cd /usr/src/dvs/dvs-apps/dvs\_run * rm \*.dmtcp |
| root@node0:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 -1/-1 1 1000 0 27342 27342 27342 27342 migr\_server  0 11 11 -1/-1 1 1000 0 27342 27342 27342 27342 migr\_client |  |
|  | * /usr/src/lkl/dmtcp/bin/dmtcp\_launch -h node1 -p 1234 ./migr\_server 0 10 |
|  | * nsenter -p -t$DC0 ./migr\_client 0 11 10 4096 100 1 & |
|  | * ./dvs\_migrate -s 0 10 |
| * ./dvs\_migrate -s 0 10 |  |
| root@node0:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 -1/-1 1 1800 0 27342 27342 27342 27342 migr\_server  0 11 11 -1/-1 1 1000 0 27342 27342 27342 27342 migr\_client | root@node1:/usr/src/dvs/dvs-apps/dvs\_run# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 801/801 1 800 20 27342 27342 27342 27342 migr\_server  0 11 11 824/2 1 8000 20 27342 27342 10 27342 migr\_client |
|  | * /usr/src/lkl/dmtcp/bin/dmtcp\_command -h node1 -p 1234 –bc * mv ckpt\_\*.dmtcp checkpoint\_migr\_server.dmtcp |
|  | * sshpass -p "root" scp -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no -r checkpoint\_migr\_server.dmtcp root@node0:/usr/src/dvs/dvs-apps/dvs\_run |
| * dmtcp\_restart checkpoint\_migr\_server.dmtcp |  |
| root@node0:~# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 722/-1 0 8 0 31438 27342 27342 27342 DMTCP:migr\_serv  0 11 11 -1/-1 1 1000 0 27342 27342 27342 27342 migr\_client |  |
|  | * ./dvs\_migrate -c 0 10 0 |
|  | root@node1:~# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 -1/-1 0 1000 0 27342 27342 27342 27342 migr\_server  0 11 11 824/2 1 0 20 27342 27342 27342 27342 migr\_client |

1. En NODE1 se arranca el coordinador de DMTCP porque allí se encuentra el servidor. Este coordinador se podría arrancar en todos los nodos del DVS.
2. En NODE1 se inicial el DVS, se configura y se inicia el DC0 en donde van a ejecutar los procesos de pruebas. Se arrancan los proxies contra el NODE0 y se adiciona el NODE0 al DC0. Esto quiere decir que desde el NODE1 se pueden referencia procesos del DC0 que se encuentren en el NODE0. Luego se setea la variable de entorno $DC0 que contiene el PID del proceso asociado al DC0. Se borran del directorio de trabajo todos los archivos de checkpoint del DMTCP.
3. En NODE0, se inicializan el DVS, el DC0, se arrancan los proxies contra NODE1 y se establece que en NODE1 pueden ejecutar procesos del DC0. También se establece que los procesos ***migr\_server*** y ***migr\_clientr*** son procesos REMOTOS (flag=1000) y se encuentran ejecutando en el NODE1 (nd=1). Luego, se borran del directorio de trabajo todos los archivos de checkpoint del DMTCP.
4. En NODE1, se arranca a través del lanzador de DMTCP la ejecución del ***migr\_server***. También se arranca utilizando ***nsenter*** la ejecución del ***migr\_client***. En este momento comienzan a transferir datos entre ambos procesos localizados en el mismo nodo.
5. En NODE1, se le notifica al DVK que el proceso ***migr\_server*** (dcid=0, endpoint=10) va a iniciar (start) un proceso de migración (flag=1800). En este momento se detienen las comunicaciones y el ***migr\_client*** queda en espera de la finalización de la migración (flag=8000). **ATENCION: Este comando de migración SIEMPRE debe ejecutarse primero en los nodos en donde se encuentran los procesos Client, de lo contrario, al no responder el Server, el Client puede dar por finalizada la comunicación**.
6. En NODE0, se le notifica al DVK que el proceso ***migr\_server*** (dcid=0, endpoint=10) va a iniciar (start) un proceso de migración.
7. En NODE1, se le solicita al coordinador de DMTCP realizar un Checkpoint bloqueante (bc) lo que provoca que se genere el archivo de checkpoint. Luego, se normaliza el nombre del archivo de checkpoint.
8. En NODE1, se transfiere el archivo de Checkpoint al NODO0 (nodo destino) utilizando ***ssh***. Para automatizar la transferencia se utiliza ***sshpass*** de tal forma de que no solicite la password del usuario de NODE0.
9. En NODE0, se rearranca el proceso ***migr\_server*** a partir de la imagen del DMTCP. El ***migr\_server*** está preparado para la migración que le solicita al DVK que lo considere migrado en el NODO1 (*dvk\_migr\_commit()*). Esta operación reinicia automáticamente las comunicaciones de todos los procesos que se estaban comunicando con el ***migr\_server*** y que fueron detenidos al realizar *dvk\_migr\_start().* Ahora, ***migr\_server*** es un proceso local del NODE0 (nd=0) que está detenido esperando recibir un mensaje de algún cliente (flag=8).
10. En NODE1, se le notifica al DVK que la migración ha resultado exitosa (*dvk\_migr\_commit()*). Ahora el ***migr\_server*** es considerado un proceso REMOTO (flag=1000) localizado en el NODE0 (nd=0). En este momento las comunicaciones continúan entre Client y Server. **ATENCION: Este comando de migración SIEMPRE debe ejecutarse primero en el nodo en donde se encuentra el procesos Server, de lo contrario, si se arranca primero en los Client el server no responderá y el Client puede dar por finalizada la comunicación.**