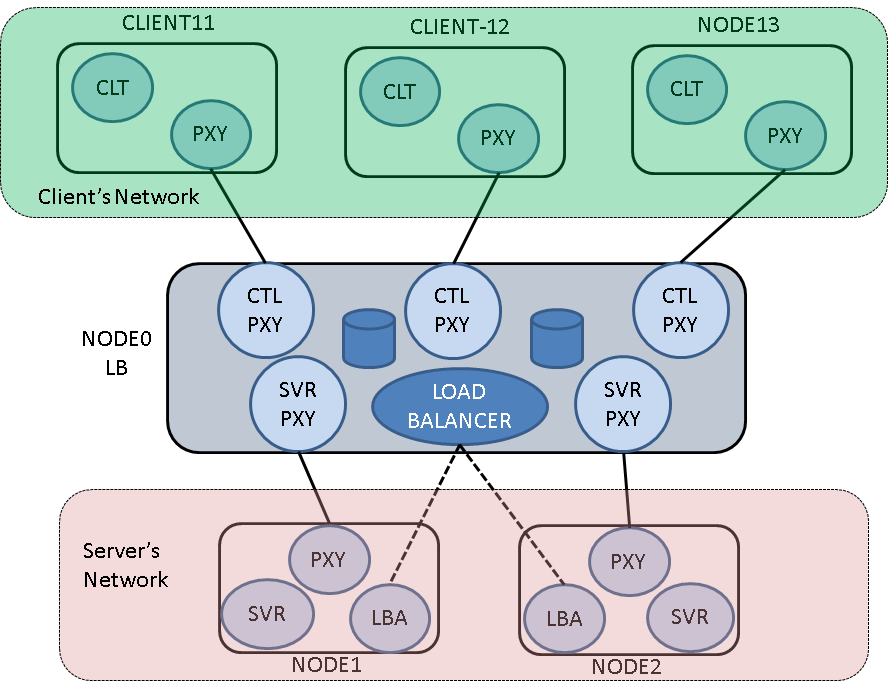
**PRUEBAS LOAD BALANCER- VERSION 2**

**TOPOLOGIA**



## Hostnames:

* **Load Balancer**: “node0”
* **Servers**: “nodeX” donde 0 < X < 11
* **Clients**: “clientYY” donde 10 < YY < 32

## Direcciones IP:

* **Load Balancer**:

RED SERVERS:

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

inet 192.168.0.100 netmask 255.255.255.0 broadcast 192.168.0.255

RED CLIENTS

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

inet 192.168.137.100 netmask 255.255.255.0 broadcast 192.168.137.255

* **Servers**: donde 0 < X < 11

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

inet 192.168.0.10X netmask 255.255.255.0 broadcast 192.168.0.25

* **Clients**: donde 10 < Y < 32

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

inet 192.168.137.1YY netmask 255.255.255.0 broadcast 192.168.137.255

## Archivo /etc/hosts

## Load Balancer

192.168.0.100 node0

192.168.0.101 node1

192.168.0.102 node2

……

192.168.137.100 client0

192.168.137.101 client1

192.168.137.102 client2

……

## Servers

192.168.0.100 node0

192.168.0.101 node1

192.168.0.102 node2

……

## Clients

192.168.137.100 client0

192.168.137.101 client1

192.168.137.102 client2

……

## Archivo /etc/spread/spread.conf

## Load Balancer y Servers

Spread\_Segment **192.168.0.255**:4803 {

node0

node1

node2

node3

node4

….

}

## Archivo de Configuración de MULTI-PROXY

## SERVERS

proxy node0 {

proxyid 0;

proto tcp;

rport 3000;

sport 300X;

compress NO;

batch NO;

autobind NO;

rname nodeX;

};

## CLIENTS

proxy client0 {

proxyid 0;

proto tcp;

rport 3000;

sport 30YY;

compress NO;

batch NO;

autobind NO;

rname clientYY;

};

## ARCHIVO DE CONFIGURACIÓN

Cuando arranca el LB el único argumento es el nombre del archivo de configuración. En este archivo se describen el Load Balancer (lb), los Servers (server), los servicios (service) y los Clients (client). En un futuro, quizás no sea necesario crear los servers ni servicios y que éstos se creen en forma automática al hacer el JOIN al grupo “DVS\_LB”.

## LOAD BALANCER

lb node0 { // ATENCION debe coincidir con el hostname

nodeid 0; // indica el nodeid dentro del cluster DVS

lowwater 30; // Uso mínimo de CPU para no eliminar server

highwater 70; // Uso maximo de CPU para crear server

period 30; // período de monitoreo

cltname client0;// hostname del LB en la red de clients

svrname node0; // hostname del LB en la red de servers

cltdev eth1; // interface del LB en la red de clients

svrdev eth0; // interface del LB en la red de servers

};

## CLIENTS Y SERVERS

server node1 {

nodeid 1;

compress NO;

batch NO;

node\_start "C:\VMware Player\vmplayer.exe start";

node\_stop "C:\ VMware Player\vmplayer.exe stop ";

node\_image "E:\NODE1\Debian 9.4.vmx";

};

client client11 {

nodeid 11;

compress NO;

batch NO;

};

## SERVICES

Los servicios requieren una mayor cantidad de parámetros tales como el Endpoint que presentarán a los Clients (ext\_ep), el rango de Endpoints que pueden utilizar los servidores para brindar esos servicios (min\_ep, max\_ep), el tipo de registración que se hará en los DVKs de los Servers (bind) al arrancar los programas servidores (prog) y el DC al que pertenece ese servicio (dcid).

# para arranque automatico de servidores

service latency {

dcid 0;

ext\_ep 10;

min\_ep 10;

max\_ep 19;

bind replica;

prog "/usr/src/dvs/dvs-apps/dvs\_lb/run\_latency.sh"

};

#para servidores previamente arrancados

service m3ftp {

dcid 0; // DCID del DC donde se ejecutará el servicio

ext\_ep 20; // Endpoint del LB hacia red de clientes

min\_ep 20; // Endpoint mínimo de los servidores

max\_ep 29; // Endpoint máximo de los servidores

};

## EJEMPLO ARCHIVO CONFIGURACION LOAD BALANCER

# this is a comment

# this is a comment

lb node0 {

nodeid 0;

lowwater 30;

highwater 70;

period 30;

cltname client0;

svrname node0;

cltdev eth1;

svrdev eth0;

};

server node1 {

nodeid 1;

compress NO;

batch NO;

node\_start "C:\Program Files (x86)\VMware\VMware Player\vmplayer.exe start";

node\_stop "C:\Program Files (x86)\VMware\VMware Player\vmplayer.exe stop ";

node\_image "E:\NODE1\Debian 9.4.vmx";

};

#dcid puede ser un numero 0-(NR\_DCS-1)

#si no se menciona dcid, entonces es ANY

#ext\_ep es el endpoint presentado a los clientes.

#min\_ep y max\_ep es el rango de endpoints en los que se pueden arrancar los services

service latency {

dcid 0;

ext\_ep 10;

min\_ep 10;

max\_ep 19;

bind replica;

prog "/usr/src/dvs/dvs-apps/dvs\_lb/run\_latency.sh"

};

service m3ftp {

dcid 0;

ext\_ep 20;

min\_ep 20;

max\_ep 29;

bind replica;

prog "/usr/src/dvs/dvs-apps/dvs\_lb/run\_m3ftpd.sh"

};

client client11 {

nodeid 11;

compress NO;

batch NO;

};

client client12 {

nodeid 12;

compress NO;

batch NO;

};

client client13 {

nodeid 13;

compress NO;

batch NO;

};

# Estructuras de Datos

## Server

Cada Server tiene un nombre (svr\_name) el cual debe estar dado de alta en el /etc/hosts del LB o bien en un DNS. Se especifica también el número de nodo (svr\_nodeid) y luego tiene una estructura de datos para su Sender proxy (svr\_spx) y otra para su Receiver Proxy (svr\_rpx).

Además posee un semáforo mutex (svr\_mutex) para proteger los campos compartidos con otros threads.

En cada server se podrá ejecutar servicios diferentes y multiples instancias de un mismo servicio, pero cada una de ellas registrada en un Endpoint diferente. El registro de los Endpoints utilizados por los servicios en cada servidor se mantiene con un bitmap (svr\_bm\_svc).

El monitor de carga del LB mantiene un registro porcentual entero de la carga (svr\_load) de un server que le es informada por el agente del server y un nivel de carga (svr\_level) que puede ser LVL\_NOTINIT, LVL\_UNLOADED, LVL\_LOADED, LVL\_SATURATED.

struct server\_s{

char \*svr\_name; // server name from configuration file

int svr\_nodeid; // server nodeid

int svr\_lbRport; // LB Receiver port

int svr\_svrRport; // Server Receiver port

int svr\_level; // Load LEVEL

int svr\_load; // CPU Load (0-100) . Value (-1) implies INVALID

int svr\_compress; // Enable LZ4 Compression 0:NO 1:YES

int svr\_batch; // Enable message batching 0:NO 1:YES

int svr\_idle\_count; // Number o Idle cycles that the server has // NOT USED YET

char \*svr\_start; // string to command which START the server NODE

char \*svr\_stop; // string to command which STOP the server NODE

char \*svr\_image; // string to command which START the server NODE

unsigned long int svr\_bm\_sts;

unsigned long int svr\_bm\_svc; // bitmap of service endpoint used

lbpx\_desc\_t svr\_spx; // Server sender proxy

lbpx\_desc\_t svr\_rpx; // Server receiver proxy

LZ4F\_errorCode\_t svr\_lz4err;

size\_t svr\_offset;

size\_t svr\_maxCsize; /\* Maximum Compressed size \*/

size\_t svr\_maxRsize; /\* Maximum Raw size \*/

\_\_attribute\_\_((packed, aligned(4)))

LZ4F\_compressionContext\_t svr\_lz4Cctx \_\_attribute\_\_((aligned(8))); /\* Compression context \*/

LZ4F\_decompressionContext\_t svr\_lz4Dctx \_\_attribute\_\_((aligned(8))); /\* Decompression context \*/

pthread\_mutex\_t svr\_mutex; // protect on change of status and load level.

pthread\_mutex\_t svr\_agent\_mtx; // controls when a server process is started or killed

pthread\_cond\_t svr\_agent\_cond;

pthread\_mutex\_t svr\_node\_mtx; // controls when a complete server node VM is started or stopped

pthread\_cond\_t svr\_node\_cond;

TAILQ\_ENTRY(server\_s) svr\_tail\_entry; /\* Tail queue. \*/

TAILQ\_HEAD(svr\_clthead,client\_s) svr\_clt\_head;

TAILQ\_HEAD(svr\_svrhead,server\_s) svr\_svr\_head;

pthread\_mutex\_t svr\_tail\_mtx; // to protect the linked list

};

typedef struct server\_s server\_t;

## Client

Cada Client tiene un nombre (clt\_name) el cual debe estar dado de alta en el /etc/hosts del LB o bien en un DNS. Se especifica también el número de nodo (clt\_nodeid) y luego tiene una estructura de datos para su Sender proxy (clt\_spx) y otra para su Receiver Proxy (clt\_rpx).

struct client\_s {

char \*clt\_name;

int clt\_nodeid;

int clt\_lbRport; // LB Receiver port

int clt\_cltRport; // Client Receiver port

int clt\_compress; // Enable LZ4 Compression 0:NO 1:YES

int clt\_batch; // Enable message batching 0:NO 1:YES

LZ4F\_errorCode\_t clt\_lz4err;

size\_t clt\_offset;

size\_t clt\_maxCsize; /\* Maximum Compressed size \*/

size\_t clt\_maxRsize; /\* Maximum Raw size \*/

\_\_attribute\_\_((packed, aligned(4)))

LZ4F\_compressionContext\_t clt\_lz4Cctx \_\_attribute\_\_((aligned(8))); /\* Compression context \*/

LZ4F\_decompressionContext\_t clt\_lz4Dctx \_\_attribute\_\_((aligned(8))); /\* Decompression context \*/

lbpx\_desc\_t clt\_spx;

lbpx\_desc\_t clt\_rpx;

pthread\_mutex\_t clt\_agent\_mtx; // controls when a server process is started or killed

pthread\_cond\_t clt\_agent\_cond;

pthread\_mutex\_t clt\_node\_mtx; // controls when a complete server node VM is started or stopped

pthread\_cond\_t clt\_node\_cond;

TAILQ\_ENTRY(client\_s) clt\_tail\_entry; /\* Tail queue. \*/

} ;

typedef struct client\_s client\_t;

## Services

Describe las características de un servicio cuyo nombre se especifica en el archivo de configuración (svc\_name).

El LB presenta en su proxy contra cada Client un Endpoint (svc\_extep) que es visto como Servidor para cada DC (svc\_dcid). Este Endpoint, no necesariamente será el que utilizará el servicio al ser arrancado en un nodo servidor. Los servicios pueden utilizar un Endpoint dentro de un rango {svc\_minep, svc\_maxep}. Además, por cada servicio se especifica el tipo de bind (replica, local, backup) y el path (en el nodo destino) del proceso que debe ejecutarse al iniciarse una sesión.

typedef struct {

char \*svc\_name; // server name from configuration file

int svc\_dcid; // server dcid

int svc\_extep; // server external endpoint

int svc\_minep; // server lower endpoint when execute the server

int svc\_maxep; // server higher endpoint when execute the server

int svc\_bind; // Which kind of bind must be done on server creation

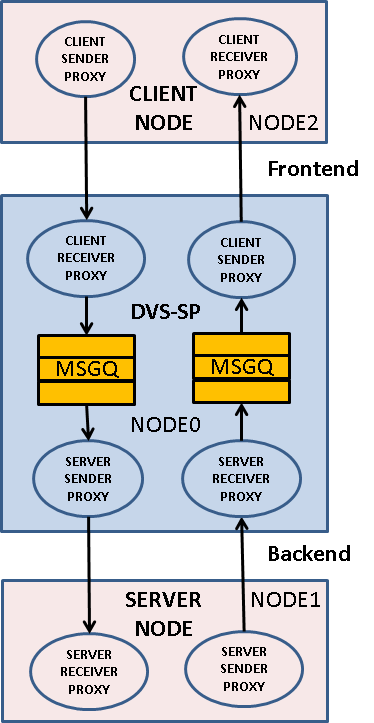
char \*svc\_prog; // Program to execute on server

}service\_t;;

## Load Balancer Proxy

Cada Client y cada Server tienen asociado un par de proxies (Sender y Receiver).

Por cada proxy se crea un thread.



El proxy Receiver de un Server recibe los mensajes de su Server correspondiente, los modifica y los reenvia al Client de la Sesión Activa a través de su Client Sender utilizando una cola de mensajes. El cmd\_t va por la cola de mensajes pero los datos se transfieren por puntero a área de memoria asignada por malloc(), o sea, no hay copia de datos entre sender y receiver.

De igual forma, el proxy Receiver de un Client recibe los mensajes de su Client correspondiente, los modifica y los reenvia al Server de la Sesión Activa o de una nueva Sesión a través de su Client Sender utilizando una cola de mensajes.

La cola de mensaje de la estructura de datos LB Proxy Descritor es solo utilizada por los Proxy Senders.

Cada Proxy tiene asociado un puntero a un buffer para el header (lbp\_header) y otro para el payload (lbp\_payload) de los mensajes recibidos (Receiver) o enviados (Sender) a través de los sockets TCP.

También contiene información de la sesiones TCP (sockets, direcciones IP, ports, etc) y contadores de mensajes Enviados/Recibidos en forma Correcta/Errónea.

/ Load Balancer Proxy Descriptor

struct lbpx\_desc\_s {

pthread\_t lbp\_thread;

proxy\_hdr\_t \*lbp\_header;

proxy\_payload\_t \*lbp\_payload; /\* uncompressed payload \*/

msgq\_buf\_t \*lbp\_mqbuf; // Server proxy sender message queue

int lbp\_mqid;

int lbp\_mqkey;

struct msqid\_ds lbp\_mqds;

int lbp\_sd; // Sender TCP socket descriptor

int lbp\_lsd; // Receiver TCP LISTEN socket descriptor

int lbp\_csd; // Sender TCP CONNECTION socket descriptor

int lbp\_port;

struct sockaddr\_in lbp\_rmtclt\_addr;

struct sockaddr\_in lbp\_rmtsvr\_addr;

#define lbp\_lclsvr\_addr lbp\_rmtsvr\_addr

struct hostent \*lbp\_rmthost;

long lbp\_msg\_ok;

long lbp\_msg\_err;

};

typedef struct lbpx\_desc\_s lbpx\_desc\_t;

## Session

Cuando un cliente inicia una comunicación contra el Client Proxy del LB se verifica si es Client tiene una sesión iniciada con un servidor designado. Para ello se comparan un conjunto de parámetros tales como el DC, sender Endpoint, destination Endpoint, etc. Además se compara el PID del Client, porque si este difiere solo en este parámetro de una sesión existente en la BD del LB, se elimina la sesión existente y se crea una nueva. Para ello se busca en el LB si hay una entrada libre en la entrada de Sesiones. Si no encuentra, retorna al cliente un mensaje EDVSNOSPC.

Si no existe una sesión, se debe asignar un Server. Esto se realiza en base a la información que el LB dispone de la carga de cada Server obtenida a partir del Monitor. Una vez seleccionado el Server, se busca en el bitmap del Server svr\_bm\_svc, un bit en 0, que significa un Endpoint libre. Los Endpoints pueden tener un valor de 0 a 32 (solo para simplificar la implementación del bitmap). Si no encuentra Endpoint libre, seguirá buscando por el server siguiente de menor carga hasta terminar la lista de Endpoints y Servers. Si no encuentra ningún Endpoint libre, entonces retorna al Client un mensaje con error EDVSNOTBIND.

Una vez asignada la Sesion, se completan sus campos en base a la información del header (incluido el subheader de vcopy) recibido desde el Client y de la información del Server obtenida de la DB del Monitor (svr\_nodeid, svr\_endpoint). Con la sesión nueva o con la previamente existente, se reenvia el mensaje recibido desde el Client (ahora con su header modificado) al Server Sender a través de su cola de mensajes.

typedef struct {

int se\_dcid;

int se\_clt\_nodeid;

int se\_clt\_ep;

int se\_clt\_PID;

int se\_lbclt\_ep;

int se\_lbsvr\_ep;

int se\_svr\_nodeid;

int se\_svr\_ep;

int se\_svr\_PID;

char \*se\_rmtcmd;

service\_t \*se\_service;

} sess\_entry\_t;

typedef struct {

int st\_nr\_sess; // # of active sessions

pthread\_mutex\_t st\_mutex; // per Table Mutex

sess\_entry\_t \*st\_tab\_ptr; // pointer to the first table element

} sess\_tab\_t;

Cuando el Server Receiver Proxy recibe un mensaje de un Server, verifica si hay una sesión existente. Si no existe sesion, y el mensaje NO ES un ACKNOWLEDGE le devuelve al Server un mensaje con error EDVSCONNREFUSED.

Si existe la sesión, pero solo difiere en el PID del Server, esto indica que es otra instancia del Server y que hay que eliminar la sesión existente.

## PARAMETROS DEL LOAD BALANCER

typedef struct {

char \*lb\_name;

int lb\_nodeid;

int lb\_lowwater; // low water load (0-100)

int lb\_highwater; // low water load (0-100)

int lb\_period; // load measurement period in seconds (1-3600)

char \*lb\_cltname; // hostname of this LB inside CLIENT NETWORK

char \*lb\_svrname; // hostname of this LB inside SERVER NETWORK

char \*lb\_cltdev; // device name of this LB inside CLIENT NETWORK i.e eth1

char \*lb\_svrdev; // device name of this LB inside SERVER NETWORK i.e eth0

pthread\_t lb\_thread;

unsigned int lb\_bm\_lbparms; // bitmap of Config Load Balancer Parameters

unsigned int lb\_bm\_svcparms;// bitmap of Config Service Parameters

unsigned int lb\_bm\_cltparms;// bitmap of Config Client Parameters

unsigned int lb\_bm\_svrparms;// bitmap of Config Server Parameters

unsigned int lb\_bm\_nodes; // bitmap of Connected nodes

unsigned int lb\_bm\_init; // bitmap initialized/active nodes

int lb\_nr\_nodes; // number of Connected server nodes

int lb\_nr\_init; // number of initialized/active server nodes

int lb\_nr\_cltpxy; // # of defined Client Proxies (from configuration file)

int lb\_nr\_svrpxy; // # of defined Server Proxies (from configuration file)

int lb\_nr\_services;// # of defined Services (from configuration file)

mailbox lb\_mbox;

int lb\_len;

char lb\_sp\_group[MAXNODENAME];

char lb\_mbr\_name[MAX\_MEMBER\_NAME];

char lb\_priv\_group[MAX\_GROUP\_NAME];

membership\_info lb\_memb\_info;

vs\_set\_info lb\_vssets[MAX\_VSSETS];

unsigned int lb\_my\_vsset\_index;

int lb\_num\_vs\_sets;

char lb\_members[MAX\_MEMBERS][MAX\_GROUP\_NAME];

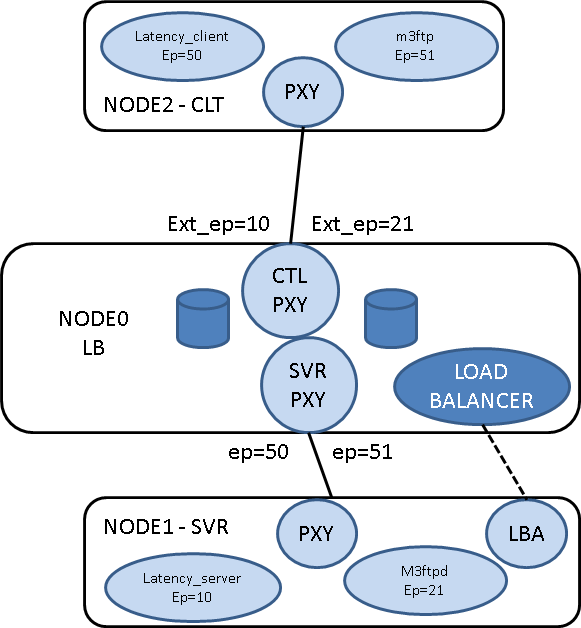
char lb\_sp\_members[MAX\_MEMBERS][MAX\_GROUP\_NAME];

int lb\_sp\_nr\_mbrs;

char lb\_mess\_in[MAX\_MESSLEN];

} lb\_t;

# PRUEBAS VERSION 1



ARCHIVO DE CONFIGURACION EN NODE0 (LOAD BALANCER)

# this is a comment

lb node0 {

nodeid 0;

lowwater 30;

highwater 70;

period 30;

cltname client0;

svrname node0;

cltdev eth1;

svrdev eth0;

};

server node1 {

nodeid 1;

};

#dcid puede ser un numero 0-(NR\_DCS-1)

#si no se menciona dcid, entonces es ANY

#ext\_ep es el endpoint presentado a los clientes.

#min\_ep y max\_ep es el rango de endpoints en los que se pueden arrancar los services

service latency {

dcid 0;

ext\_ep 10;

min\_ep 10;

max\_ep 19;

};

service m3ftp {

dcid 0;

ext\_ep 20;

min\_ep 20;

max\_ep 29;

};

client client11 {

nodeid 11;

};

ATENCION, UTILIZAR MULTI-PROXY

* SIN AUTO BIND
* SIN COMPRESION
* SIN BATCH
* CON PID AGREGADO

/usr/src/dvs/dvk-proxies/multi\_proxy multi\_proxy$lcl.cfg > /dev/shm/node$lcl.txt 2> /dev/shm/error$lcl.txt &

PARA LAS PRUEBA REMOVER:

* DEBUG DEL DVK
* DEBUG DE tests.sh
* DEBUG DE LA LIB
* DEBUG DE LOS PROXIES
* DEBUG DEL LB
* DEBUG DE LATENCY\_CLIENT Y LATENCY\_SERVER
* DEBUG DE FTPD Y FTP

Crear un archivo de 1 MB, otro de 10 MB y otro de 100M

dd bs=1024 count=1024 </dev/urandom > file-1M.dat

dd bs=1024 count=10240 </dev/urandom > file-10M.dat

dd bs=1024 count=102400 </dev/urandom > file-100M.dat

***TEST LOCALES***

|  |  |
| --- | --- |
| **SEQ** | **NODE1** |
| 1 | cd /usr/src/dvs/dvk-tests/ |
| 2 | ./tests.sh 1 0  . /dev/shm/DC0.sh |
| 3 | ./latency\_server 0 10 > latency0.out 2> latency0.err &  ./latency\_server 0 11 > latency1.out 2> latency1.err &  ./latency\_server 0 12 > latency2.out 2> latency2.err & |
| 4 | cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 21 > m3ftpd0.out 2> m3ftpd0.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 22 > m3ftpd1.out 2> m3ftpd1.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 23 > m3ftpd2.out 2> m3ftpd2.err &  cat /proc/dvs/DC0/procs |
| 5 | cd /usr/src/dvs/dvk-tests/ |
| 6 | ./latency\_client 0 50 10 1000 > lat0.out &  ./latency\_client 0 51 10 1000 > lat1.out &  ./latency\_client 0 52 10 1000 > lat2.out & |
| 7 | cd /usr/src/dvs/dvs-apps/m3ftp/  ./m3ftp -g 0 60 21 file-1M.dat file0-1M.dat > gftp0.out &  ./m3ftp -g 0 61 22 file-1M.dat file1-1M.dat > gftp1.out &  ./m3ftp -g 0 62 23 file-1M.dat file2-1M.dat > gftp2.out & |
|  | ./m3ftp -p 0 60 21 file0a-1M.dat file0-1M.dat > pftp0.out &  ./m3ftp -p 0 61 22 file1a-1M.dat file1-1M.dat > pftp1.out &  ./m3ftp -p 0 62 23 file2a-1M.dat file2-1M.dat > pftp2.out & |
|  | ./m3ftp -g 0 60 21 file-10M.dat file0-10M.dat > gftp0.out &  ./m3ftp -g 0 61 22 file-10M.dat file1-10M.dat > gftp1.out &  ./m3ftp -g 0 62 23 file-10M.dat file2-10M.dat > gftp2.out & |
|  | ./m3ftp -p 0 60 21 file0a-10M.dat file0-10M.dat > pftp0.out &  ./m3ftp -p 0 61 22 file1a-10M.dat file1-10M.dat > pftp1.out &  ./m3ftp -p 0 62 23 file2a-10M.dat file2-10M.dat > pftp2.out & |
|  | ./m3ftp -g 0 60 21 file-100M.dat file0-100M.dat > gftp0.out & |
|  | ./m3ftp -p 0 60 21 file0a-100M.dat file0-100M.dat > pftp0.out & |

***TEST REMOTOS***

***En este test, node0 no se usa como Load Balancer sino como CLIENT***

|  |  |  |
| --- | --- | --- |
| **SEQ** | **NODE1(SERVER)** | **NODE0(CLIENT)** |
|  | cd /usr/src/dvs/dvk-tests/ | cd /usr/src/dvs/dvk-tests/ |
|  | ./tests.sh 1 0 | ./tests.sh 0 0 |
|  | **ESPERAR QUE TERMINE** | **ESPERAR QUE TERMINE** |
|  | netstat –nat  . /dev/shm/DC0.sh | netstat –nat  . /dev/shm/DC0.sh |
|  | ./test\_rmtbind 0 **50** 0 latclient0  ./test\_rmtbind 0 **51** 0 latclient1  ./test\_rmtbind 0 **52** 0 latclient2  ./test\_rmtbind 0 **60** 0 m3ftp0  ./test\_rmtbind 0 **61** 0 m3ftp1  ./test\_rmtbind 0 **62** 0 m3ftp2  ./test\_rmtbind 0 **63** 0 m3ftp3  ./test\_rmtbind 0 **64** 0 m3ftp4  ./test\_rmtbind 0 **65** 0 m3ftp5  cat /proc/dvs/DC0/procs | ./test\_rmtbind 0 **10** 1 latency\_server  ./test\_rmtbind 0 **11** 1 latency\_server  ./test\_rmtbind 0 **12** 1 latency\_server  ./test\_rmtbind 0 **21** 1 m3ftpd  ./test\_rmtbind 0 **22** 1 m3ftpd  ./test\_rmtbind 0 **23** 1 m3ftpd  ./test\_rmtbind 0 **24** 1 m3ftpd  ./test\_rmtbind 0 **25** 1 m3ftpd  ./test\_rmtbind 0 **26** 1 m3ftpd  cat /proc/dvs/DC0/procs |
|  | ./latency\_server 0 10 > latency0.out 2> latency0.err &  ./latency\_server 0 11 > latency1.out 2> latency1.err & ./latency\_server 0 12 > latency2.out 2> latency2.err & |  |
|  | cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 21 > m3ftpd0.out 2> m3ftpd0.err &  ./m3ftpd 0 22 > m3ftpd1.out 2> m3ftpd1.err &  ./m3ftpd 0 23 > m3ftpd2.out 2> m3ftpd2.err &  ./m3ftpd 0 24 > m3ftpd3.out 2> m3ftpd3.err &  ./m3ftpd 0 25 > m3ftpd4.out 2> m3ftpd4.err &  ./m3ftpd 0 26 > m3ftpd5.out 2> m3ftpd5.err &  cat /proc/dvs/DC0/procs |  |
|  |  | ./latency\_client 0 50 10 100 > lat0.out &  ./latency\_client 0 51 10 100 > lat1.out &  ./latency\_client 0 52 10 100 > lat2.out & |
|  |  | cd /usr/src/dvs/dvs-apps/m3ftp/  ./m3ftp -g 0 60 21 file-1M.dat file0-1M.dat > gftp0.out &  ./m3ftp -g 0 61 22 file-1M.dat file1-1M.dat > gftp1.out &  ./m3ftp -g 0 62 23 file-1M.dat file2-1M.dat > gftp2.out & |
|  |  | ./m3ftp -p 0 60 21 file0-1M.dat file0-1M.dat > pftp0.out &  ./m3ftp -p 0 61 22 file1-1M.dat file1-1M.dat > pftp1.out &  ./m3ftp -p 0 62 23 file2-1M.dat file2-1M.dat > pftp2.out & |
|  |  | ./m3ftp -g 0 60 21 file-10M.dat file0-10M.dat > gftp0.out &  ./m3ftp -g 0 61 22 file-10M.dat file1-10M.dat > gftp1.out &  ./m3ftp -g 0 62 23 file-10M.dat file2-10M.dat > gftp2.out &  ./m3ftp -g 0 63 24 file-10M.dat file3-10M.dat > gftp3.out &  ./m3ftp -g 0 64 25 file-10M.dat file4-10M.dat > gftp4.out &  ./m3ftp -g 0 65 26 file-10M.dat file5-10M.dat > gftp5.out & |
|  |  | ./m3ftp -p 0 60 21 file0-10M.dat file0-10M.dat > pftp0.out &  ./m3ftp -p 0 61 22 file1-10M.dat file1-10M.dat > pftp1.out &  ./m3ftp -p 0 62 23 file2-10M.dat file2-10M.dat > pftp2.out & |
|  |  | ./m3ftp -g 0 60 21 file-100M.dat file0-100M.dat > gftp0.out &  ./m3ftp -g 0 61 22 file-100M.dat file1-100M.dat > gftp1.out &  ./m3ftp -g 0 62 23 file-100M.dat file2-100M.dat > gftp2.out &  ./m3ftp -g 0 63 24 file-100M.dat file3-100M.dat > gftp3.out &  ./m3ftp -g 0 64 25 file-100M.dat file4-100M.dat > gftp4.out &  ./m3ftp -g 0 65 26 file-100M.dat file5-100M.dat > gftp5.out & |
|  |  | ./m3ftp -p 0 60 21 file0-100M.dat file0-100M.dat > pftp0.out & |

***TEST REMOTOS CON LOAD BALANCER SINGLE SERVER***

|  |  |  |  |
| --- | --- | --- | --- |
| **SEQ** | **NODE0 (LB)** | **NODE1(SERVER)** | **CLIENT11(CLIENT)** |
|  | mkdir /var/run/spread  /usr/local/sbin/spread -c /etc/spread.conf > /dev/shm/spread.txt & |  |  |
| 1 | cd /usr/src/dvs/dvs-apps/dvs\_lb/ |  |  |
| 2 | ./lb\_dvs lb\_dvs.cfg > lb\_dvs.out 2> lb\_dvs.err & |  |  |
| 3 |  | cd /usr/src/dvs/dvk-tests/ | cd /usr/src/dvs/dvk-tests/ |
| 4 |  | ./tests.sh 1 0 | ./tests.sh 11 0 |
|  |  | **ESPERAR QUE TERMINE** | **ESPERAR QUE TERMINE** |
| 5 |  | netstat –nat  . /dev/shm/DC0.sh | netstat –nat  . /dev/shm/DC0.sh |
| 6 |  | ./test\_rmtbind 0 **50** 0 latclient0  ./test\_rmtbind 0 **51** 0 latclient1  ./test\_rmtbind 0 **52** 0 latclient2  ./test\_rmtbind 0 **60** 0 m3ftp0  ./test\_rmtbind 0 **61** 0 m3ftp1  ./test\_rmtbind 0 **62** 0 m3ftp2  ./test\_rmtbind 0 **63** 0 m3ftp3  ./test\_rmtbind 0 **64** 0 m3ftp4  ./test\_rmtbind 0 **65** 0 m3ftp5  cat /proc/dvs/DC0/procs | ./test\_rmtbind 0 **10** 0 latency\_server  ./test\_rmtbind 0 **11** 0 latency\_server  ./test\_rmtbind 0 **12** 0 latency\_server  ./test\_rmtbind 0 **20** 0 m3ftpd0  ./test\_rmtbind 0 **21** 0 m3ftpd1  ./test\_rmtbind 0 **22** 0 m3ftpd2  ./test\_rmtbind 0 **23** 0 m3ftpd3  ./test\_rmtbind 0 **24** 0 m3ftpd4  ./test\_rmtbind 0 **25** 0 m3ftpd5  cat /proc/dvs/DC0/procs |
| 7 |  | ./latency\_server 0 **10** > latency0.out 2> latency0.err &  ./latency\_server 0 **11** > latency1.out 2> latency1.err & ./latency\_server 0 **12** > latency2.out 2> latency2.err & |  |
| 8 |  | cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **20** > m3ftpd0.out 2> m3ftpd0.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **21** > m3ftpd1.out 2> m3ftpd1.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **22** > m3ftpd2.out 2> m3ftpd2.err &  ./m3ftpd 0 **23** > m3ftpd3.out 2> m3ftpd3.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **24** > m3ftpd4.out 2> m3ftpd4.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **25** > m3ftpd5.out 2> m3ftpd5.err &  cat /proc/dvs/DC0/procs |  |
| 9 |  |  | ./latency\_client 0 **50 10** 100 > lat0.out &  ./latency\_client 0 **51 11** 100 > lat1.out &  ./latency\_client 0 **52 12** 100 > lat2.out & |
|  |  |  | cd /usr/src/dvs/dvs-apps/m3ftp/  ./m3ftp -g 0 **60 20** file-1M.dat file0-1M.dat > gftp0.out &  ./m3ftp -g 0 **61 21** file-1M.dat file1-1M.dat > gftp1.out &  ./m3ftp -g 0 **62 22** file-1M.dat file2-1M.dat > gftp2.out & |
|  |  |  | ./m3ftp -p 0 **60 20** file0-1M.dat file0-1M.dat > pftp0.out &  ./m3ftp -p 0 **61 21** file1-1M.dat file1-1M.dat > pftp1.out &  ./m3ftp -p 0 **62 22** file2-1M.dat file2-1M.dat > pftp2.out & |
|  |  |  | ./m3ftp -g 0 **60 20** file-10M.dat file0-10M.dat > gftp0.out &  ./m3ftp -g 0 **61 21** file-10M.dat file1-10M.dat > gftp1.out &  ./m3ftp -g 0 **62 22** file-10M.dat file2-10M.dat > gftp2.out & |
|  |  |  | ./m3ftp -p 0 **60 20** file0-10M.dat file0-10M.dat > pftp0.out &  ./m3ftp -p 0 **61 21** file1-10M.dat file1-10M.dat > pftp1.out &  ./m3ftp -p 0 **62 22** file2-10M.dat file2-10M.dat > pftp2.out & |
|  |  |  | ./m3ftp -g 0 60 20 file-100M.dat file0-100M.dat > gftp0.out &  ./m3ftp -g 0 61 21 file-100M.dat file1-100M.dat > gftp1.out &  ./m3ftp -g 0 62 22 file-100M.dat file2-100M.dat > gftp2.out &  ./m3ftp -g 0 63 23 file-100M.dat file3-100M.dat > gftp3.out &  ./m3ftp -g 0 64 24 file-100M.dat file4-100M.dat > gftp4.out &  ./m3ftp -g 0 65 25 file-100M.dat file5-100M.dat > gftp5.out & |
|  |  |  | ./m3ftp -p 0 60 20 file0-100M.dat file0-100M.dat > pftp0.out & |

***TEST REMOTOS CON LOAD BALANCER, MULTIPLE SERVER, MULTIPLE CLIENT***

|  |  |  |  |
| --- | --- | --- | --- |
| **SEQ** | **NODE0 (LB)** | **SERVER** | **CLIENT** |
|  | mkdir /var/run/spread  /usr/local/sbin/spread -c /etc/spread.conf > /dev/shm/spread.txt & |  |  |
| 1 | cd /usr/src/dvs/dvs-apps/dvs\_lb/ |  |  |
| 2 | ./lb\_dvs lb\_dvs.cfg > lb\_dvs.out 2> lb\_dvs.err & |  |  |
| 3 |  | cd /usr/src/dvs/dvk-tests/ | cd /usr/src/dvs/dvk-tests/ |
| 4 |  | ./tests.sh 1 0 | ./tests.sh 11 0 |
|  |  | **ESPERAR QUE TERMINE** | **ESPERAR QUE TERMINE** |
| 5 |  | netstat –nat  . /dev/shm/DC0.sh | netstat –nat  . /dev/shm/DC0.sh |
| 6 |  | ./test\_rmtbind 0 **50** 0 latclient0  ./test\_rmtbind 0 **51** 0 latclient1  ./test\_rmtbind 0 **52** 0 latclient2  ./test\_rmtbind 0 **53** 0 latclient3  ./test\_rmtbind 0 **54** 0 latclient4  ./test\_rmtbind 0 **55** 0 latclient5  ./test\_rmtbind 0 **60** 0 m3ftp0  ./test\_rmtbind 0 **61** 0 m3ftp1  ./test\_rmtbind 0 **62** 0 m3ftp2  ./test\_rmtbind 0 **63** 0 m3ftp3  ./test\_rmtbind 0 **64** 0 m3ftp4  ./test\_rmtbind 0 **65** 0 m3ftp5  cat /proc/dvs/DC0/procs | ./test\_rmtbind 0 **10** 0 latency\_server  ./test\_rmtbind 0 **11** 0 latency\_server  ./test\_rmtbind 0 **12** 0 latency\_server  ./test\_rmtbind 0 **20** 0 m3ftpd0  ./test\_rmtbind 0 **21** 0 m3ftpd1  ./test\_rmtbind 0 **22** 0 m3ftpd2  ./test\_rmtbind 0 **23** 0 m3ftpd3  ./test\_rmtbind 0 **24** 0 m3ftpd4  ./test\_rmtbind 0 **25** 0 m3ftpd5  cat /proc/dvs/DC0/procs |
| 7 |  | ./latency\_server 0 **10** > latency0.out 2> latency0.err &  ./latency\_server 0 **11** > latency1.out 2> latency1.err & ./latency\_server 0 **12** > latency2.out 2> latency2.err & ./latency\_server 0 **13** > latency3.out 2> latency0.err &  ./latency\_server 0 **14** > latency4.out 2> latency1.err & ./latency\_server 0 **15** > latency5.out 2> latency2.err & |  |
| 8 |  | cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **20** > m3ftpd0.out 2> m3ftpd0.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **21** > m3ftpd1.out 2> m3ftpd1.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **22** > m3ftpd2.out 2> m3ftpd2.err &  ./m3ftpd 0 **23** > m3ftpd3.out 2> m3ftpd3.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **24** > m3ftpd4.out 2> m3ftpd4.err &  cd /usr/src/dvs/dvs-apps/m3ftp  ./m3ftpd 0 **25** > m3ftpd5.out 2> m3ftpd5.err &  cat /proc/dvs/DC0/procs |  |
| 9 |  |  | ./latency\_client 0 **50 10** 100 > lat0.out &  ./latency\_client 0 **51 11** 100 > lat1.out &  ./latency\_client 0 **52 12** 100 > lat2.out & |
|  |  |  | cd /usr/src/dvs/dvs-apps/m3ftp/  ./m3ftp -g 0 **60 20** file-1M.dat file0-1M.dat > gftp0.out &  ./m3ftp -g 0 **61 21** file-1M.dat file1-1M.dat > gftp1.out &  ./m3ftp -g 0 **62 22** file-1M.dat file2-1M.dat > gftp2.out & |
|  |  |  | ./m3ftp -p 0 **60 20** file0-1M.dat file0-1M.dat > pftp0.out &  ./m3ftp -p 0 **61 21** file1-1M.dat file1-1M.dat > pftp1.out &  ./m3ftp -p 0 **62 22** file2-1M.dat file2-1M.dat > pftp2.out & |
|  |  |  | ./m3ftp -g 0 **60 20** file-10M.dat file0-10M.dat > gftp0.out &  ./m3ftp -g 0 **61 21** file-10M.dat file1-10M.dat > gftp1.out &  ./m3ftp -g 0 **62 22** file-10M.dat file2-10M.dat > gftp2.out & |
|  |  |  | ./m3ftp -p 0 **60 20** file0-10M.dat file0-10M.dat > pftp0.out &  ./m3ftp -p 0 **61 21** file1-10M.dat file1-10M.dat > pftp1.out &  ./m3ftp -p 0 **62 22** file2-10M.dat file2-10M.dat > pftp2.out & |
|  |  |  | ./m3ftp -g 0 60 20 file-100M.dat file0-100M.dat > gftp0.out &  ./m3ftp -g 0 61 21 file-100M.dat file1-100M.dat > gftp1.out &  ./m3ftp -g 0 62 22 file-100M.dat file2-100M.dat > gftp2.out &  ./m3ftp -g 0 63 23 file-100M.dat file3-100M.dat > gftp3.out &  ./m3ftp -g 0 64 24 file-100M.dat file4-100M.dat > gftp4.out &  ./m3ftp -g 0 65 25 file-100M.dat file5-100M.dat > gftp5.out & |
|  |  |  | ./m3ftp -p 0 60 20 file0-100M.dat file0-100M.dat > pftp0.out & |

# PRUEBAS VERSION 2

Se desarrolló el script dvs/dvk-tests/test\_lb.sh

Este script se arranca automáticamente al iniciar Linux con el script

/etc/init.d/dvs

Luego se hizo

chmod 755 /etc/init.d/dvs

update-rc.d dvs defaults

Este script, identifica al nodo en base a su hostname. En la variable $lb se define cual es el número de nodo que será el Load Balancer.

* Si el nombre es serverXX: se configurará como un server con ID=XX
* Si el nombre es clientXX: se configurará como un client con ID=XX

Para los nodos diferentes al LB, se arranca el multiproxy para conectarse con el LB.

Si el nodo es:

* Server: Se arranca el agente correspondiente
* Load Balancer: Se arranca el load balancer

Es decir, una vez finalizado el script de arranque los 3 tipos de nodos (LB, SERVERS, CLIENTS) deberían estar conectados por sus proxies y el Server con su agente arrancado.

Para realizar las pruebas se pueden arrancar los procesos servers a mano o bien configurar en el archivo de configuración, cual es el programa que debe arrancar, generalmente un script, por ejemplo

prog "/usr/src/dvs/dvs-apps/dvs\_lb/run\_latency.sh"

prog "/usr/src/dvs/dvs-apps/dvs\_lb/run\_m3ftpd.sh"

Estos scripts reciben como parámetros desde el LB ***<dcid> <svr\_ep> <clt\_node> <clt\_ep>.***

De tal modo que el server se arranque en el <dcid> con Endpoint <svr\_ep> y se haga un dvk\_rmtbind del Endpoint cliente <clt\_ep> en el nodo <clt\_node>

|  |  |
| --- | --- |
| **NODE** |  |
| NODE0 | **Se verifica el arranque del LB**  root@node0:~# ps -ef | grep dvs  root 590 1 0 09:25 ? 00:00:00 ./lb\_dvs lb\_dvs.cfg  **Se verifica la conexion de los proxies**  root@node0:~# netstat -natp | grep dvs  tcp 0 0 192.168.0.100:3001 0.0.0.0:\* LISTEN 590/./lb\_dvs  tcp 0 0 192.168.0.100:3002 0.0.0.0:\* LISTEN 590/./lb\_dvs  tcp 0 0 192.168.137.100:3011 0.0.0.0:\* LISTEN 590/./lb\_dvs  tcp 0 0 192.168.137.100:3012 0.0.0.0:\* LISTEN 590/./lb\_dvs  tcp 0 0 192.168.137.100:3013 0.0.0.0:\* LISTEN 590/./lb\_dvs  tcp 0 0 192.168.0.100:50618 192.168.0.101:3000 ESTABLISHED 590/./lb\_dvs  tcp 0 0 192.168.0.100:3001 192.168.0.101:49674 ESTABLISHED 590/./lb\_dvs  tcp 0 1 192.168.0.100:43828 192.168.0.102:3000 SYN\_SENT 590/./lb\_dvs  tcp 0 0 192.168.137.100:54304 192.168.137.111:3000 ESTABLISHED 590/./lb\_dvs  tcp 0 1 192.168.137.100:56188 192.168.137.113:3000 SYN\_SENT 590/./lb\_dvs  tcp 0 0 192.168.137.100:3011 192.168.137.111:60462 ESTABLISHED 590/./lb\_dvs  tcp 0 1 192.168.137.100:33260 192.168.137.112:3000 SYN\_SENT 590/./lb\_dvs  tcp 0 1 192.168.137.100:33250 192.168.137.112:3000 SYN\_SENT 590/./lb\_dvs |
| NODE1 | **Se verifica el arranque del LB AGENT**  root@node1:~# ps -ef | grep agent  root 722 1 0 09:25 ? 00:00:00 ./lb\_agent  **Se verifica la conexion contra el LB**  root@node1:~# netstat -natp | grep multi  tcp 0 0 192.168.0.101:3000 0.0.0.0:\* LISTEN 606/multi\_proxy  tcp 0 0 192.168.0.101:3000 192.168.0.100:50618 ESTABLISHED 606/multi\_proxy  tcp 0 0 192.168.0.101:49674 192.168.0.100:3001 ESTABLISHED 606/multi\_proxy |
| CLIENT11 | **root@client11:~# netstat -natp | grep multi**  tcp 0 0 192.168.137.111:3000 0.0.0.0:\* LISTEN 586/multi\_proxy  tcp 0 0 192.168.137.111:60462 192.168.137.100:3011 ESTABLISHED 586/multi\_proxy  tcp 0 0 192.168.137.111:3000 192.168.137.100:54304 ESTABLISHED 586/multi\_proxy |
| CLIENT11 | **Se bindea el server y luego se prueban varios clients desde diferentes endpoints**  root@client11:~# cd /usr/src/dvs/dvk-tests/  root@client11:/usr/src/dvs/dvk-tests# ./test\_rmtbind 0 10 0 latency\_server  dvk\_rmtbind latency\_server with p\_nr=10 to DC0 on node=0  root@client11:/usr/src/dvs/dvk-tests# ./latency\_client 0 50 10 5  root@client11:/usr/src/dvs/dvk-tests# ./latency\_client 0 51 10 5  root@client11:/usr/src/dvs/dvk-tests# ./latency\_client 0 52 10 5  root@client11:/usr/src/dvs/dvk-tests# ./latency\_client 0 53 10 5  root@client11:/usr/src/dvs/dvk-tests# ./latency\_client 0 54 10 5  root@client11:/usr/src/dvs/dvk-tests# ./latency\_client 0 55 10 5 |
| NODE1 | **Se verifica el arranque de los servers**  root@node1:~# ps -ef | grep latency  root 986 1 0 09:47 ? 00:00:00 ./latency\_server 0 10  root 992 1 0 09:48 ? 00:00:00 ./latency\_server 0 11  root 997 1 0 09:48 ? 00:00:00 ./latency\_server 0 12  root 1003 1 0 09:48 ? 00:00:00 ./latency\_server 0 13  root 1008 1 0 09:48 ? 00:00:00 ./latency\_server 0 14  root 1014 1 0 09:48 ? 00:00:00 ./latency\_server 0 15  **Se verifican los Endpoints**  root@node1:~# cat /proc/dvs/DC0/procs  DC pnr -endp -lpid/vpid- nd flag misc -getf -sndt -wmig -prxy name  0 10 10 986/986 1 8 20 31438 27342 27342 27342 latency\_server  0 11 11 992/992 1 8 20 31438 27342 27342 27342 latency\_server  0 12 12 997/997 1 8 20 31438 27342 27342 27342 latency\_server  0 13 13 1003/1003 1 8 20 31438 27342 27342 27342 latency\_server  0 14 14 1008/1008 1 8 20 31438 27342 27342 27342 latency\_server  0 15 15 1014/1014 1 8 20 31438 27342 27342 27342 latency\_server  0 50 50 -1/-1 0 1000 0 27342 27342 27342 27342 clt\_0\_0\_50  0 51 51 -1/-1 0 1000 0 27342 27342 27342 27342 clt\_0\_0\_51  0 52 52 -1/-1 0 1000 0 27342 27342 27342 27342 clt\_0\_0\_52  0 53 53 -1/-1 0 1000 0 27342 27342 27342 27342 clt\_0\_0\_53  0 54 54 -1/-1 0 1000 0 27342 27342 27342 27342 clt\_0\_0\_54  0 55 55 -1/-1 0 1000 0 27342 27342 27342 27342 clt\_0\_0\_55  **Se verifican las estadísticas**  root@node1:~# cat /proc/dvs/DC0/stats  DC pnr -endp -lpid/vpid- nd --lsnt-- --rsnt-- -lcopy-- -rcopy-- name  0 10 10 986/986 1 0 5 0 0 latency\_server  0 11 11 992/992 1 0 5 0 0 latency\_server  0 12 12 997/997 1 0 5 0 0 latency\_server  0 13 13 1003/1003 1 0 5 0 0 latency\_server  0 14 14 1008/1008 1 0 5 0 0 latency\_server  0 15 15 1014/1014 1 0 5 0 0 latency\_server  0 50 50 -1/-1 0 0 0 0 0 clt\_0\_0\_50  0 51 51 -1/-1 0 0 0 0 0 clt\_0\_0\_51  0 52 52 -1/-1 0 0 0 0 0 clt\_0\_0\_52  0 53 53 -1/-1 0 0 0 0 0 clt\_0\_0\_53  0 54 54 -1/-1 0 0 0 0 0 clt\_0\_0\_54  0 55 55 -1/-1 0 0 0 0 0 clt\_0\_0\_55 |