# Capítulo 4

*Este capítulo se centra en la implementación, se detallan y explican los trabajos necesarios para la instalación y configuración de Solaris Containers.*

## 4. IMPLEMENTACIÓN

Siguiendo la metodología que se ha descrito en el capítulo anterior, se decidió incluir en capítulos separados los temas de Implementación y Evaluación, debido a la complejidad y a la cantidad de información que se presenta en cada una de ellas. El presente capítulo abarca la etapa posterior al diseño desde la adquisición de los equipos, la adecuación del espacio, hasta la liberación de los servicios, de acuerdo a la infraestructura con que se cuente. Para el caso de este trabajo, la parte de adquisición y montado físico del equipo no aplica, sin embargo en la gran mayoría de los proyectos es necesario considerarlo.

El capítulo está dividido en diferentes subtemas de acuerdo con la configuración específica que llevaremos, los cuales son:

- Instalación del Sistema Operativo base
- Configuración del Sistema Operativo base
    - Configuración de Red
    - Configuración de Hardening
    - Configuración de Espejo de Discos (RAID 1)
- Configuración de Solaris Containers
    - Configuración del Pool de Recursos para las Zonas
    - Definición de Zonas
- Instalación de Webstack
    - Servidor de Aplicaciones
    - Servidor de Base de Datos
- Configuración de Webstack

Estos subtemas abarcan la configuración llevada a cabo, de tal forma que sea posible entender progresivamente cada uno de los puntos de implementación.


Figura 4.1 Servidor T2000

## 4.1 Instalación del Sistema Operativo base

El sistema operativo Solaris 10 puede ser instalado mediante la red o desde un medio (cd/dvd). Los requerimientos mínimos para su instalación son:

- Memoria Ram mínima de 256 MB
- Espacio en disco duro 5GB
- Procesador 200 MHz o superior

A continuación se resume la instalación de Solaris 10, cabe mencionar que el proceso detallado de la instalación del sistema operativo base se encuentra en el Anexo 2.

El Programa de instalación de Solaris ofrece una opción mediante interfaz gráfica de usuario (GUI) y otra mediante texto, ya sea desde una conexión al puerto serial o desde un monitor conectado al servidor. De igual forma, se pueden utilizar otras opciones de Instalación como son el Jumpstart y el Flash Archive. En la primera podemos definir las características del sistema a instalar a partir de otro sistema existente; y el método de instalación Flash Archive consiste en instalar y configurar un sistema base del que se realiza un respaldo Flash Archive, éste respaldo se usa para instalar el sistema operativo en uno o varios sistemas más.

Para el caso del servidor Sun Fire T2000 la instalación inicia en modo de texto, viendo la salida por medio del puerto serial, ya que no se cuenta con interfaz grafica. La primera pantalla es la de configuración del sistema y requiere que se elija el idioma del teclado y de la Interfaz, seleccionamos **Inglés.**

En las pantallas siguientes proporcionamos el nombre del servidor el cual debe ser único, en este caso se le asignó el nombre **UXSOP001** así como algunos de los parámetros de red: indicamos que pertenece a una subred, así como su máscara de subred **255.255.255.0**, establecemos que se trata de una asignación de IP manual y no por **Dynamic Host Configuration Protocol** (**DHCP**), proporcionamos la dirección IP la cual también debe ser única **192.9.198.190**, deshabilitamos la opción de *Ipv6* ya que no vamos a utilizar ese protocolo, posteriormente especificamos la dirección IP del Router que vamos a utilizar, **192.9.198.1.** (Figura 4.2).

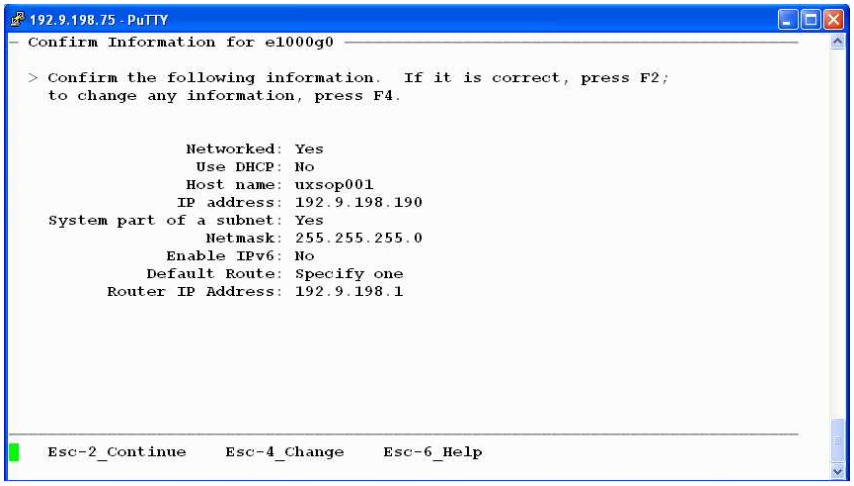



Figura 4.2 Instalación del Servidor

A continuación configuramos las políticas de seguridad, mediante la activación de **Kerberos** o de la **seguridad estándar de UNIX**. *Kerberos* es un sistema de autenticación utilizado para comprobar la identidad de un usuario o máquina, mientras que la *seguridad*

*estándar de UNIX*, no sólo abarca la autenticación sino también otros aspectos de seguridad que involucran la red, passwords, sistemas de archivos, entre otros. Es por esta razón que elegimos la Seguridad Estándar de UNIX. El siguiente paso es indicar si el servidor tendrá o no un servicio denominado **Name Service**, ya sea NIS, DNS o LDAP en nuestra configuración seleccionamos **NONE,** ya que el servidor no tendrá ninguno de los servicios antes mencionados y posteriormente seleccionamos **Use the NFSv4 domain derived by the system** con lo cual se genera un nombre de dominio de forma automática el cual se utiliza para la resolución interna. La siguiente sección requiere la selección de información relacionada con la ubicación, en nuestro caso en Time Zone es **Americas**, en Country **Mexico y Central Time** para la ciudad de México (Figura 4.3).

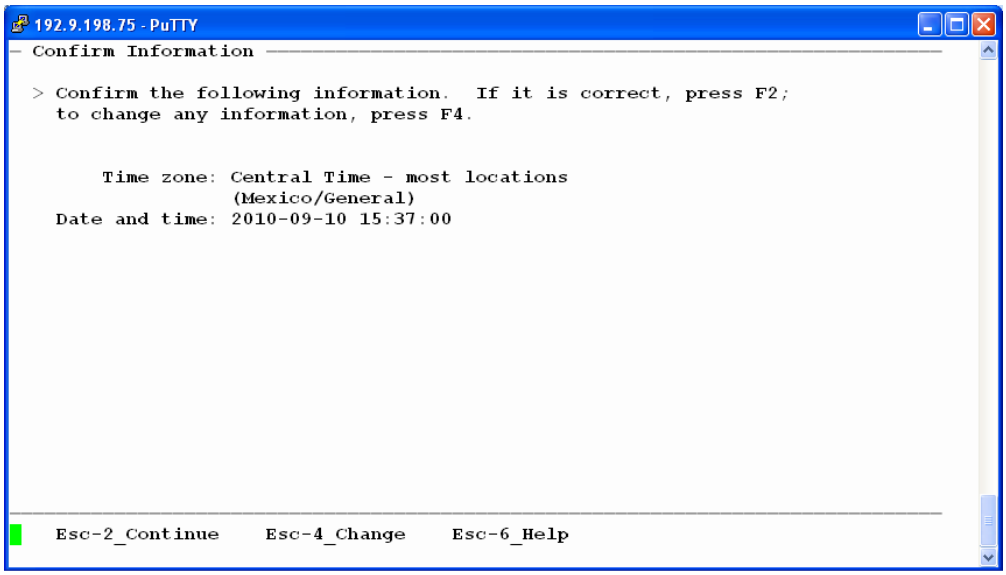



Figura 4.3 Instalación del Servidor

El usuario root es el que tiene el mayor nivel de privilegios y es por esto que el instalador solicita el **password de root**. Finalmente establecemos la configuración para **Remote Services Enabled**, que son todos los servicios de red para clientes remotos, por razones de seguridad seleccionamos la opción de **NO**, donde deshabilitamos estos servicios con la excepción de Secure Socket Layer (SSH), esto se realiza con la finalidad de reducir la superficie de ataque mediante la desactivación de los servicios de red que no se van a utilizar.

De ser necesario se configuran los servicios de forma independiente después de la finalización de la instalación, una vez hecho esto se muestra el progreso de la instalación **System Identificación Complete.**

Una vez que el sistema inicializa procedemos a seleccionar el sistema de archivos que se va a utilizar, se selecciona **UFS** que es el nativo en Solaris para la instalación del Sistema Operativo y el **ZFS** se utilizará en las zonas ya que ofrece un mayor espacio para los archivos, mejora enormemente la administración y cuenta con un buen nivel de seguridad de datos, a continuación preferimos la opción de **Entire Distribution plus OEM support** del menú de software, esta opción incluye controladores de hardware adicionales incluso para hardware que se va incorporar posteriormente a la instalación, además de los paquetes para el grupo de software de distribución completa, a continuación seleccionamos el disco duro para la instalación del software **c1t0d0**. En el paso siguiente, configuramos el sistema de archivos, se ofrecen las opciones de **Auto Layout,** esta opción genera una sola partición de disco de arranque completo, mientras que la opción **Manual Layout** permite asignar los recursos necesarios para cada partición; preferimos la segunda opción ya que es más granular y con esto se distribuyen mejor los recursos, lo que da lugar a un mejor y mayor eficiencia.

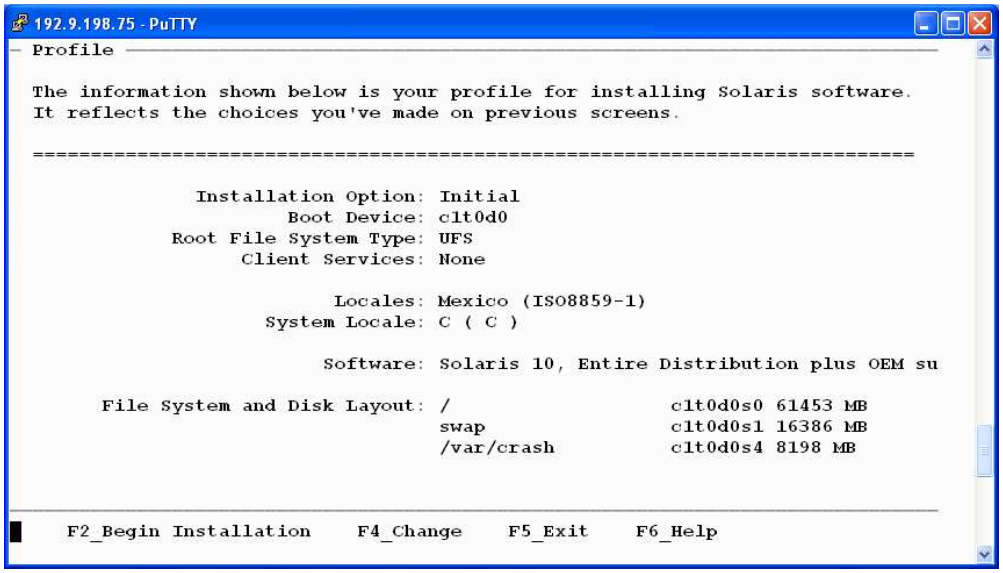



Figura 4.4 Configuración del disco

Una vez hecho esto comienza la instalación donde se nos informan los cambios que se realizaron en disco duro (Figura 4.4).

Posteriormente, el instalador muestra el número de bytes instalados al igual que los faltantes, así como una breve descripción de la herramienta que se encuentra instalando y en la parte inferior aparece el porcentaje de instalación en una escala de 0 a 100, donde 100 % es la finalización del proceso de instalación Figura (4.5).

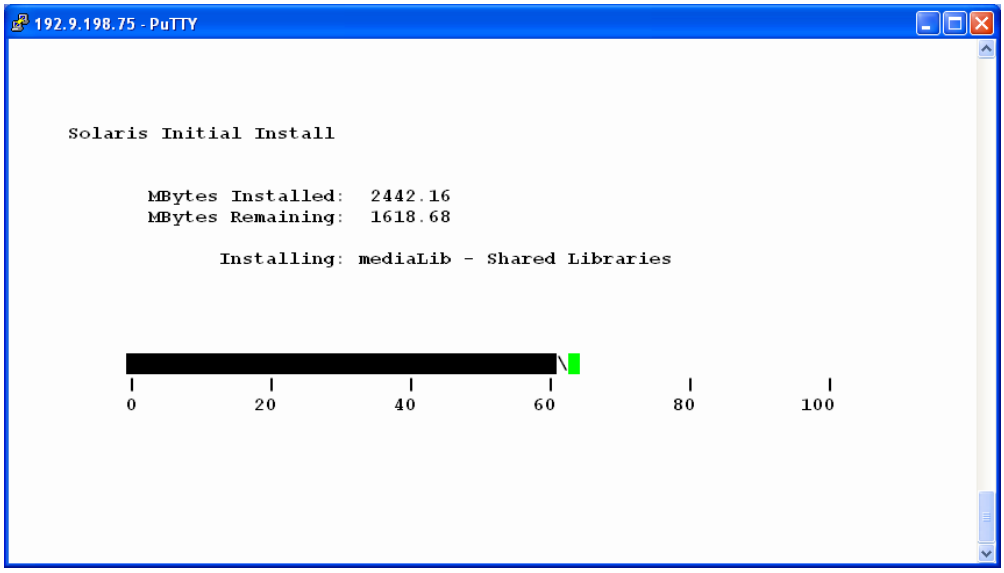



Figura 4.5 Proceso de Instalación

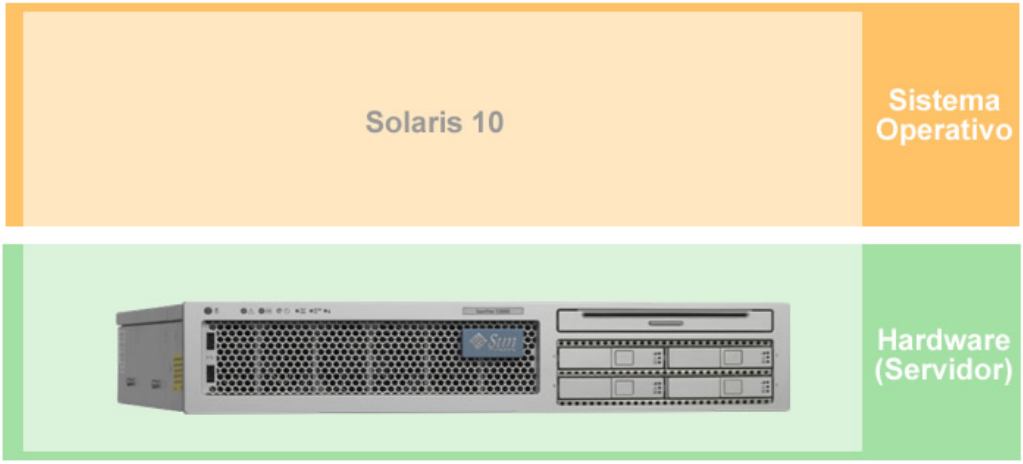



Figura 4.6 Sistema Operativo Base

## 4.2. Configuración del Sistema Operativo base

Posterior a la instalación del Sistema Operativo, en donde se instalan los archivos base y configuraciones mínimas que permiten operar el servidor, es necesario realizar configuraciones con las que se cumplan los requerimientos de diseño.

De acuerdo al diseño establecido para nuestro trabajo de Tesis, la configuración del Sistema Operativo base incluye los siguientes rubros, los cuales se detallan más adelante en este subtema:

- Configuración de Red
- Configuración de Hardening
- Configuración de Espejo de Discos

### 4.2.1. Red

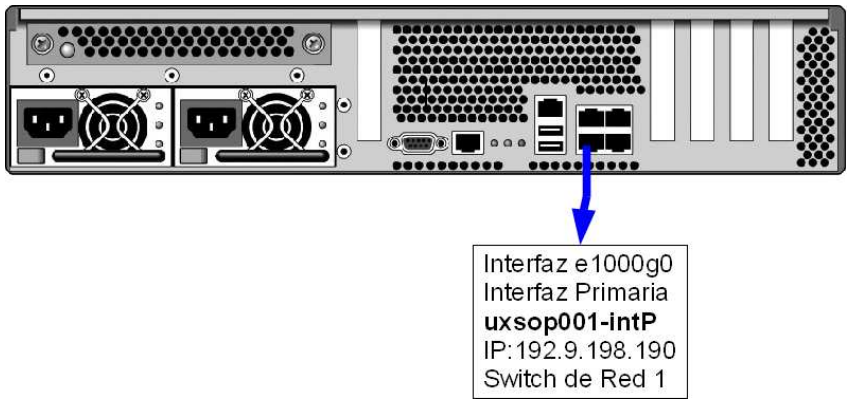



Figura 4.7 Configuración Inicial de Red

Las interfaces de red deben cumplir con ciertos requisitos para poder configurarse utilizando IPMP, deben tener direcciones MAC únicas es por ello que en caso de tener una dirección MAC compartida como sucede en equipos SPARC es importante cambiar su configuración, de tal forma que cada interfaz tenga su propia MAC, adicionalmente deben utilizar el mismo medio y compartir el mismo vinculo IP.

A continuación se describe brevemente la configuración IPMP del tipo **Activa-Reserva** de las interfaces de red del servidor Sun Fire T2000.

En el servidor UXSOP001 mediante el uso del comando **cat**, listamos el contenido del archivo */etc/hosts*, verificamos que este archivo contenga IP, nombre *FQDN (Fully Qualified Domain Name)* del servidor. Después comprobamos las interfaces disponibles, con el parámetro **show-dev** del comando **dladm**, se listan todos lo interfaces disponibles y su estado. Observamos que se tienen 2 interfaces de red, **e1000g0** y **e1000g2**, respectivamente tienen una velocidad de 100Mbps y están conectadas al mismo vínculo IP.

```
uxsop001 / # dladm show-dev
e1000g0         link: up        speed: 100   Mbps       duplex: full
e1000g1         link: unknown   speed: 0     Mbps       duplex: half
e1000g2         link: up        speed: 100   Mbps       duplex: full
e1000g3         link: unknown   speed: 0     Mbps       duplex: half
```

La interfaz e1000g2 no se encuentra en línea, es por ello que utilizamos el comando ifconfig con el parámetro **plumb**, lo que pone en línea la interfaz y con el parámetro *–a* verificamos el estatus de la interfaz, una vez hecho esto establecemos los parámetros de red de la interfaz: dirección IP y su máscara de subred.

```
uxsop001 / # ifconfig e1000g2 plumb
uxsop001 / # ifconfig -a
lo0:   flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL>   mtu   8232
index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.9.198.190 netmask ffffff00 broadcast 192.9.198.255
        ether 0:3:ba:d8:fb:ce
e1000g2: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 0.0.0.0 netmask 0
        ether 0:3:ba:d8:fb:d0
uxsop001 / # ifconfig e1000g2 192.9.198.130 netmask 255.255.255.0 up
```

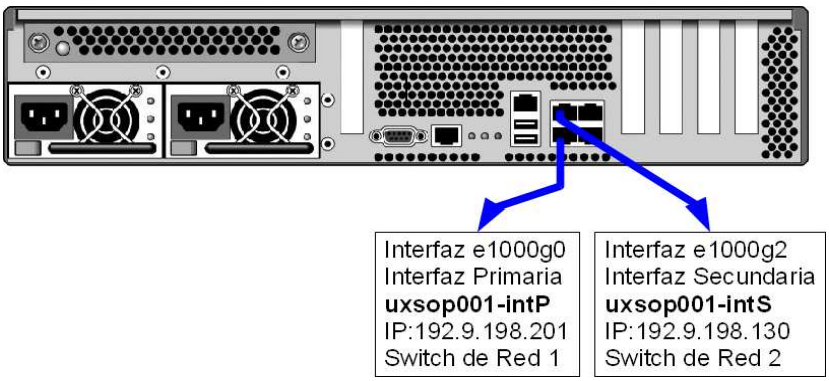



Figura 4.8 Configuración de las interfaces de red

La configuración IPMP del tipo Activa-Reserva requiere que una interfaz sea la primaria la cual llamaremos uxsop001-intP y la otra de reserva uxsop001-intS, es por esto que establecemos estos parámetros en el archivo ***/etc/hosts***.

```
uxsop001 / # cat /etc/hosts
#
# Internet host table
#
::1     localhost
127.0.0.1       localhost
192.9.198.190   uxsop001        uxsop001.tesis.com      loghost

#Direccion IP Interfaz principal
192.9.198.201   uxsop001-intP

#Direccion IP Interfaz secundaria
192.9.198.130   uxsop001-intS
```

La interfaz principal **uxsop001-intP** y secundaria **uxsop001-intS** las configuramos para que pertenezcan al mismo grupo, el grupo tiene un nombre para su distinción y en nuestro

caso le llamaremos **ipmp0**, posteriormente se establece una dirección de prueba, esto con la finalidad de que realice el proceso de la detección de fallos.

Este proceso se basa en sondeos en una interfaz específica mediante los parámetros -**failover deprecated up**, todo esto se realiza en el directorio *etc* en los archivos *hostname.e1000g0* y *hostname.e1000g2*, ya que de lo contrario se perderá esta configuración al reinicio del equipo. Únicamente en la interfaz principal definimos la interfaz lógica mediante **addif**.

```
uxsop001 / # cat /etc/hostname.e1000g0
uxsop001-intP netmask + broadcast + deprecated -failover group ipmp0 up \
addif uxsop001 netmask + broadcast + up

uxsop001 / # cat /etc/hostname.e1000g2
uxsop001-intS netmask + broadcast + deprecated -failover group ipmp0 up
```

Finalmente reiniciamos el servidor que ya tiene configurado IPMP, una vez que el equipo ha iniciado verificamos la correcta configuración de las interfaces, con lo que se garantiza la redundancia en la red (Figura 4.9).

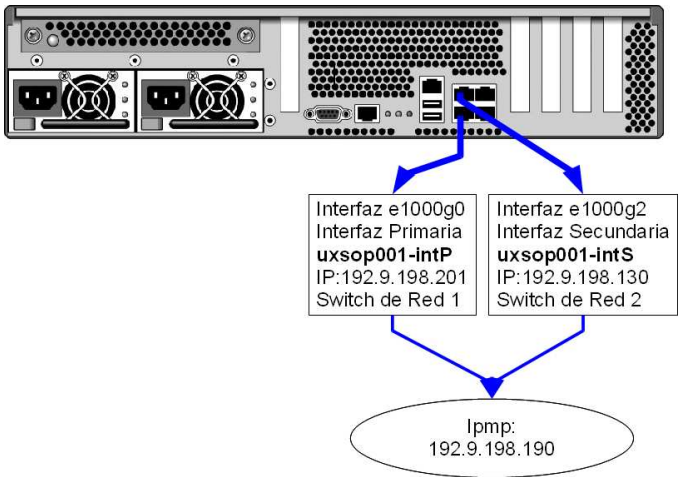



Figura 4.9 Configuración IPMP

### 4.2.2. Hardening

Después de la instalación Solaris 10, se deben realizar ciertas tareas de configuración y personalización del sistema operativo, al ser éste el host base para los Solaris Containers, es necesario preparar el sistema para prevenir ataques de seguridad.

El objetivo es remover servicios vulnerables o innecesarios, eliminar los huecos de seguridad que puedan existir y mejorar la configuración de los servicios ofrecidos para

estar preparados ante amenazas que puedan atentar contra la aplicación ofrecida, así como resguardar la información que se maneja en el servidor y en los Solaris Containers.

**Seguridad física**

El servidor Sun Fire T2000 cuenta con una memoria RAM no volátil (NVRAM) denominada OpenBoot PROM (OBP), cuenta con una interfaz de software que se encarga de verificar el estado del hardware para después arrancar el sistema operativo, ayudar al diagnóstico de problemas y obtener información sobre el servidor, todas estas operaciones son por medio de línea de comandos.

Es posible acceder a la OBP por medio de la controladora del sistema (system controller) mandando la sentencia de ***break***, o bien terminando la ejecución del sistema operativo. Esto representa un problema de seguridad ya que alguien con acceso al servidor conectado a la controladora del sistema, podría enviar la sentencia de ***break*** intencionalmente o por error, lo que ocasionaría la detención de todos los procesos que se encuentren ejecutándose en el sistema operativo, además de proporcionar total control sobre el sistema y sus parámetros de configuración a nivel de hardware.

Para evitar esta situación deshabilitamos la función de ***break***. Se coloca en el archivo de configuración */etc/system* la directiva: **KEYBOARD_ABORT = disable**. Evitando de esta manera la detención del sistema operativo enviando la función break.

A nivel de OBP se configuro el parámetro security-mode, en modo ***"command-secure"***, para que al reiniciar el servidor, y se intente el arranque con un dispositivo que no sea el indicado en la configuración inicial solicite la contraseña proporcionada.

**Optimización de parámetros de red**

Los parámetros de red que se cargan al momento en que inicia Solaris 10, necesitan ser modificados para optimizar las funciones de red. Estos parámetros por default presentan los siguientes valores:

```
ip_respond_to_echo_broadcast 1
ip_forward_directed_broadcasts 0
ip_strict_dst_multihoming 0
ip_ignore_redirect 0
ip_forwarding 0
ip_forward_src_routed 0
ip_ire_arp_interval 1200000
arp_cleanup_interval 300000
ip_respond_to_timestamp 0
ip_respond_to_timestamp_broadcast 0
```

Al modificar estos parámetros se pueden prevenir ataques y errores, por ejemplo: errores por un ping de broadcast, se pueden bloquear los paquetes de broadcast hacia la red, prevenir **Spoofing**, evitar un redireccionamiento de IP, eliminar paquetes de enrutamiento, se puede redefinir el tiempo de expiración de las tablas de ARP y broadcast para la sintonización de reloj y fecha ICMP, entre otros.

Para modificar los valores se crea un script de inicio en la ruta */etc/rc2.d/S99netconfig*.

```
#!/bin/ksh
/usr/sbin/ndd -set /dev/ip ip_respond_to_echo_broadcast 0
/usr/sbin/ndd -set /dev/ip ip_forward_directed_broadcasts 0
/usr/sbin/ndd -set /dev/ip ip_strict_dst_multihoming 1
/usr/sbin/ndd -set /dev/ip ip_ignore_redirect 1
/usr/sbin/ndd -set /dev/ip ip_forwarding 0
/usr/sbin/ndd -set /dev/ip ip_forward_src_routed 0
/usr/sbin/ndd -set /dev/ip ip_ire_arp_interval 60000
/usr/sbin/ndd -set /dev/arp arp_cleanup_interval 60000
/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp 0
/usr/sbin/ndd -set /dev/ip ip_respond_to_timestamp_broadcast 0
```

**Configuración de valores del kernel**

Para optimizar el nivel de seguridad del Sistema Operativo Solaris, existen diversos ajustes que se pueden llevar a cabo a nivel de kernel. En Solaris, existe el archivo */etc/system* que contiene parámetros específicos del kernel.

Algunos de los ajustes que deben de considerarse para mantener un buen nivel de seguridad, son por ejemplo:

- Lista de Ejecutables

Algunos programas que atacan las vulnerabilidades de seguridad de los sistemas (también conocidos como "*exploits*"), toman ventaja de la lista de ejecutables de kernel del

sistema para atacarlo. Estos programas intentan sobreescribir partes especiales de un ejecutable privilegiado de kernel (es decir, un programa de kernel que cuente con permisos para controlar el sistema) para poder controlarlo. En Solaris, estos "*exploits*" pueden evadirse haciendo de la lista de ejecutables de kernel, una lista *no ejecutable*. Agregando las siguientes líneas al archivo */etc/system* se puede evitar una vulnerabilidad del sistema:

```
set noexec_user_stack=1
set noexec_user_stack_log=1
```

Con estas variables definidas el sistema reportará en un archivo log cualquier intento de ejecución no permitido así como el nombre de usuario que intento realizar la ejecución. Esto permitirá que Solaris cuente con más resistencia ante este tipo de ataques.

- Tamaño de los archivos Core

Este tipo de archivos se encargan de guardar información sobre programas o procesos que durante su ejecución terminaron de forma inesperada, por lo que se utilizan para la detección de errores, debido al gran numero de información que pueden contener es necesario limitar el tamaño de estos ya que pueden en un momento dado ocupar demasiado espacio en disco. Esto lo hacemos con el parámetro.

```
set sys:coredumpsize = 0
```

**Deshabilitar servicios innecesarios**

Los servicios en un sistema operativo Solaris 10 son ofrecidos por el demonio **inetd** encargado de manejar las conexiones que llegan al servidor, cuando se realiza una petición a un servicio de red, el demonio **inetd** interpreta la solicitud y lo envía al servicio encargado de esa solicitud. En Solaris 10 el archivo que contiene la descripción de los servicios de red es */etc/inetd.conf* y el archivo con el número de puerto que utilizan los servicios es */etc/services*. Si estos servicios de red no son imprescindibles para el funcionamiento del sistema, se puede prescindir de ellos comentando con un signo "#" su declaración en los archivos descritos, por lo que al recibir una petición de un servicio que nosotros no hemos habilitado el servidor no responde a la petición por no encontrar su descripción.

Otros servicios de red son ofrecidos por demonios independientes que se ejecutan por medio de scripts declarados en los directorios de los diferentes niveles de ejecución del sistema al momento en que el servidor inicia.

Los directorios de niveles de ejecución del sistema (run level) contienen los scripts con los procedimientos para inicializar o detener los demonios de los servicios de red a inicializar cuando el sistema cambia de estado a alguno de los niveles de ejecución.

Los directorios y el Nivel les de ejecución de Solaris 10, se muestran en la tabla 4.1:

| Directorio | Nivel de ejecución |
|---|---|
| /etc/rcS.d | single user |
| /etc/rc0.d | Shutdown |
| /etc/rc1.d | Start |
| /etc/rc2.d | multi-user |
| /etc/rc3.d | multi-user (por default) |
| /etc/rc4.d | multi-user (sin utilizar) |
| /etc/rc5.d | shutdown y poweroff |
| /etc/rc6.d | shutdown y reboot |

Para evitar que se ejecuten los scripts al momento de iniciar el sistema, se detiene la ejecución del demonio con el argumento "***stop"*** y en los directorios */etc/rc2.d* y */etc/rc3.d* se renombran los scripts, que son identificados con la letra "S" seguido de un número y el nombre del servicio.

```
/etc/rc3.d/S16boot.server stop
mv /etc/rc3.d/S16boot.server  /etc/rc3.d/NO_S16boot.server

/etc/rc3.d/S50apache stop
mv /etc/rc3.d/S50apache /etc/rc3.d/NO_S50apache

/etc/rc2.d/S99dtlogin stop
mv /etc/rc2.d/S99dtlogin /etc/rc2.d/NO_S99dtlogin

/etc/rc3.d/S90samba stop
mv /etc/rc3.d/S90samba /etc/rc3.d/NO_S90samba

/etc/rc3.d/S80mipagent stop
mv /etc/rc3.d/S80mipagent /etc/rc3.d/NO_S80mipagent

/etc/rc3.d/S52imq stop
mv /etc/rc3.d/S52imq /etc/rc3.d/NO_S52imq

/etc/rc3.d/S16boot.server stop
mv /etc/rc2.d/S70uucp /etc/rc2.d/NO_S70uucp

/etc/rc2.d/S47pppd stop
mv /etc/rc2.d/S47pppd /etc/rc2.d/NO_S47pppd
```

Posteriormente en caso de requerir la ejecución de alguno de estos servicios bastaría con pasarle al script renombrado el argumento *"**start"***.

### Actualización del Sistema Operativo

Las actualizaciones de Solaris 10 son necesarias para corregir problemas, errores o debilidades en seguridad del Sistema Operativo. Se proveen por medio de parches, un parche es un conjunto de archivos y directorios que actualizan o corrigen archivos y directorios existentes pero que se ha detectado que puedan presentar errores o ser susceptibles a ataques de seguridad.

Los parches que son necesarios para la correcta ejecución del sistema son recomendados y los de seguridad para prevenir los ataques o vulnerabilidades. Estos son distribuidos al público en general por Internet, sin embargo, existen parches para errores específicos los cuales solo están disponibles para clientes con un contrato de mantenimiento.

Una vez descargados el conjunto de parches recomendados y el conjunto de parches de seguridad se transfieren a la carpeta */var/tmp* del servidor para su instalación (Figura 4.10).

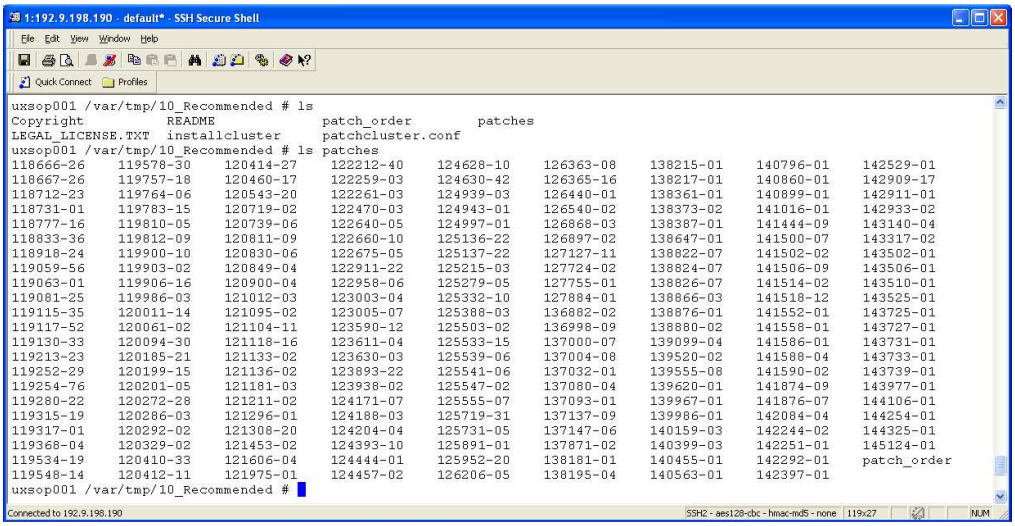


Figura 4.10 Lista de parches

Dentro del conjunto de parches encontraremos cada parche de forma individual, el cual reconocemos por un numero identificador ***"patch id"*** seguido de un numero de revisión

del parche. Así como un archivo de descripción README en el cual se especifica la forma de instalación y los problemas que corrige.

Para la instalación del conjunto de parches recomendados ejecutamos el archivo ***installcluster***, que instala uno por uno cada parche y si este no es requerido por Solaris 10, no lo instala y continua con el siguiente parche.

Puede ser que no sea necesario instalar cierto parche debido a que se cuenta con una versión actualizada o no contamos con la versión del software el cual presenta el error.

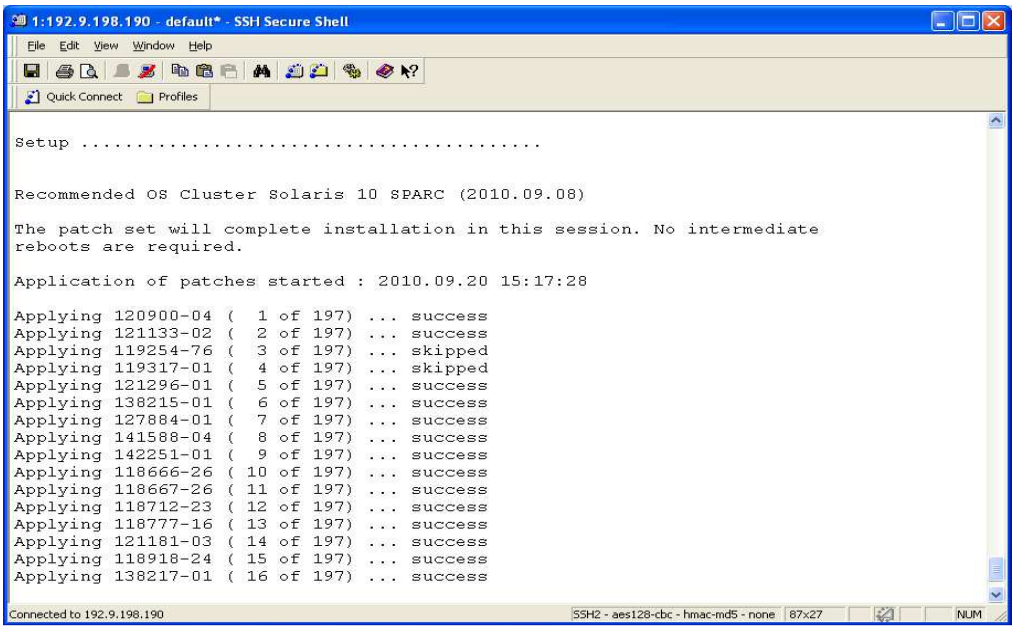



Figura 4.11 Instalación de parches

**Usuarios**

Solaris 10 contiene varios usuarios por defecto, se tienen que modificar los privilegios y atributos o en su caso borrar ya que no son necesarias para la ejecución, con el objetivo de fortalecer el aseguramiento del sistema.

Las cuentas de usuarios eliminadas, con el uso del comando ***passmgmt*** con la opción -d son:

- smtp
- nuucp
- listen

Los usuarios se encuentran eliminados, así como toda referencia a ellos en los arvhivos ***/etc/passwd*** y ***/etc/shadow***.

El resto de los usuarios por defecto son modificados para agregar más seguridad, ya que no cuentan con un Shell ni con un password asignado. Para evitar que estas cuentas de usuario sean utilizadas se utiliza el comando ***passwd*** con la opción –l para bloquear cada una de ellas. Para impedir que puedan ejecutar comandos se asigna a cada una de las cuentas el Shell /bin/false.

La Tabla 4.2 muestra la modificación de usuarios de Sistema Operativo

| Usuario | Comentario | Estatus inicial | Estatus modificado | Shell inicial | Shell actual |
|---|---|---|---|---|---|
| Daemon | Sin comentario | NL (Non-login) | Bloqueada | Sin Shell | /bin/false |
| Bin | Sin comentario | NL (Non-login) | Bloqueada | Sin Shell | /bin/false |
| Sys | Sin comentario | NL (Non-login) | Bloqueada | Sin Shell | /bin/false |
| Adm | Admin | NL (Non-login) | NL (Non-login) | Sin Shell | /bin/false |
| Lp | Line Printer Admin | NL (Non-login) | Bloqueada | Sin Shell | /bin/false |
| Uucp | Uucp Admin | NL (Non-login) | Bloqueada | Sin Shell | /bin/false |
| Smmsp | Sendmail message Submission Program | NL (Non-login) | Bloqueada | Sin Shell | /bin/false |
| Gdm | GDM reserved UID | Bloqueada | Bloqueada | Sin Shell | /bin/false |
| Webservd | WebServer reserved UID | Bloqueada | Bloqueada | Sin Shell | /bin/false |
| Postgres | PostgreSQL Reserved UID | NL (Non-login) | Bloqueada | /usr/bin/ pfksh | /bin/false |
| Svcstag | Service Tag UID | Bloqueada | Bloqueada | Sin Shell | /bin/false |
| Nobody | NFS Anonymous Access User | Bloqueada | Bloqueada | Sin Shell | /bin/false |
| Noaccess | No Access User | Bloqueada | Bloqueada | Sin Shell | /bin/false |
| Nobody4 | SunOS 4.x NFS Anonymous Access User | Bloqueada | Bloqueada | Sin shell | /bin/false |

Las cuentas de usuarios con el estatus NL (Non-Login) son cuentas que deben existir en el Sistema Operativo, por ejemplo para reservar un identificador de usuario (UID), sin tener acceso interactivo para la ejecución de comandos. Sin embargo, estas cuentas pueden ejecutar tareas programadas por medio de ***cron*** y ***at*** al ser bloqueadas son restringidas por completo.

***Cron*** y ***At*** ejecutan procesos en segundo plano en el Sistema Operativo de forma programada, por lo que cualquier usuario con permisos puede calendarizar la ejecución de un proceso de forma automática. Son de gran ayuda para los administradores del sistema, puesto que ayudan en las tareas de administración.

Pero un usuario no administrativo podría atacar el sistema al dejar corriendo de forma infinita un proceso en segundo plano. Al momento de la instalación de Solaris 10, se crean los archivos *cron.deny* y *at.deny* en la ruta /etc/cron.d/:

```
uxsop001 /etc/cron.d # ls -lrt
total 6
-rw-r--r--   1 root     sys           17 Jan 21  2005 queuedefs
-rw-r--r--   1 root     sys           40 Sep  3  2009 cron.deny
-rw-r--r--   1 root     sys           40 Sep  3  2009 at.deny
prw-------   1 root     root           0 Oct  9 20:06 FIFO
```

Por lo que se agregan a estos archivos los usuarios no administrativos, dejando con permisos de ejecución a “root” y “adm”. Ya que estos ejecutan tareas programadas de administración, importantes para la recolección de información del funcionamiento del equipo.

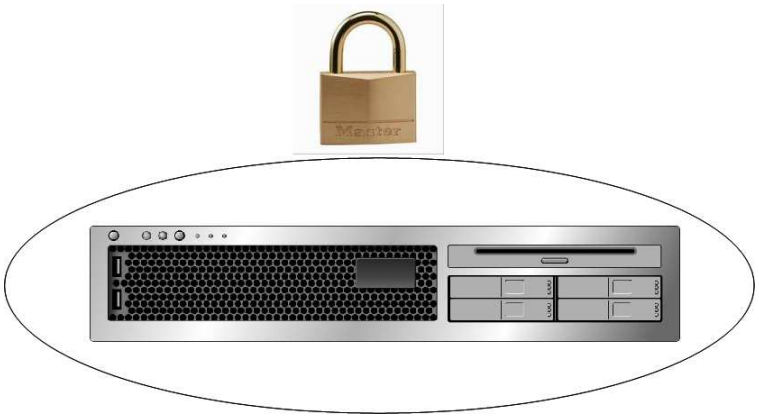


Figura 4.12 Servidor Asegurado

### 4.2.3. Configuración de Espejo de Discos (RAID 1)

Para la configuración de espejo de discos del sistema operativo, se requiere contar con dos discos idénticos en cuanto a tamaño, partición y distribución de espacio en los sistemas de archivos.

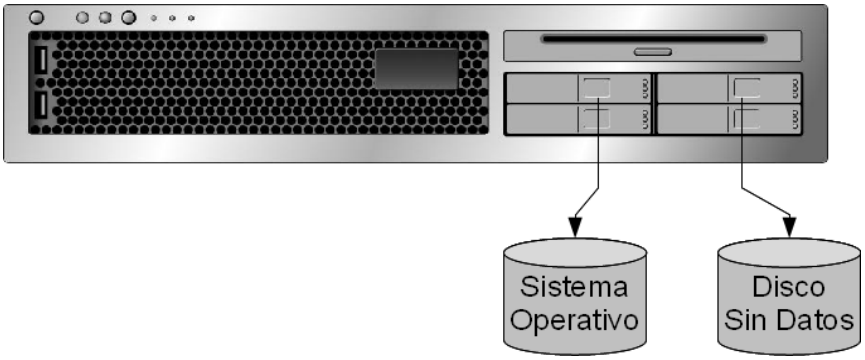



Figura 4.13 Configuración inicial de discos

En primera instancia es necesario crear una base de datos de replicas, las replicas es donde se va almacenar la información y configuración de los dispositivos del espejo. La base de datos se creó utilizando la partición 7 de cada uno de los discos.

```
uxsop001 / # metadb -a -f -c 3 c1t0d0s7 c1t1d0s7
uxsop001 / # metadb -i
        flags           first blk       block count
     a m  p  luo        16              8192            /dev/dsk/c1t0d0s7
     a    p  luo        8208            8192            /dev/dsk/c1t0d0s7
     a    p  luo        16400           8192            /dev/dsk/c1t0d0s7
     a    p  luo        16              8192            /dev/dsk/c1t1d0s7
     a    p  luo        8208            8192            /dev/dsk/c1t1d0s7
     a    p  luo        16400           8192            /dev/dsk/c1t1d0s7
 r - replica does not have device relocation information
 o - replica active prior to last mddb configuration change
 u - replica is up to date
 l - locator for this replica was read successfully
 c - replica's location was in /etc/lvm/mddb.cf
 p - replica's location was patched in kernel
 m - replica is master, this is replica selected as input
 W - replica has device write errors
 a - replica is active, commits are occurring to this replica
 M - replica had problem with master blocks
 D - replica had problem with data blocks
 F - replica had format problems
 S - replica is too small to hold current data base
 R - replica had device read errors
```

Se cuenta con dos discos de 146 Gb, c1t0d0 es el primer disco y c1t1d0 es el disco espejo, los cuales se encuentran particionado y se muestran en la tabla 4.3 las Particiones de los discos de arranque.

| Partición | Uso | Tamaño |
|---|---|---|
| 0 | / | 60.01GB |
| 1 | Swap | 16.00GB |
| 4 | /var | 8.01GB |
| 7 | bd de replicas | 258.38MB |

La configuración del espejo se hace convirtiendo cada partición de cada disco en un metadispositivo (un subespejo) mediante el comando **metainit** con la opción –f, especificando que es un subespejo de una copia (uno a uno) y la partición del disco. La nomenclatura para nombrar los metadispositivos es la letra “d” seguida del numero de disco (1 ó 2) y la partición a configurar (1, 2 ó 4).

```
uxsop001 / # metainit -f d10 1 1 c1t0d0s0
d10: Concat/Stripe is setup
uxsop001 / # metainit -f d20 1 1 c1t1d0s0
d20: Concat/Stripe is setup
uxsop001 / # metainit -f d11 1 1 c1t0d0s1
d11: Concat/Stripe is setup
uxsop001 / # metainit -f d21 1 1 c1t1d0s1
d21: Concat/Stripe is setup
uxsop001 / # metainit -f d14 1 1 c1t0d0s4
d14: Concat/Stripe is setup
uxsop001 / # metainit -f d24 1 1 c1t1d0s4
d24: Concat/Stripe is setup
```

Para las particiones s0, s1 y s4 se hace el espejo, lo que se logra creando los volúmenes d0, d1 y d4. Con el comando **metainit** creamos los subespejos, pero en esta ocasión con la opción –m indicando que es un espejo del subespejo del primer disco.

```
uxsop001 / # metainit d0 -m d10
d0: Mirror is setup
uxsop001 / # metainit d1 -m d11
d1: Mirror is setup
uxsop001 / # metainit d4 -m d14
d4: Mirror is setup
```

Una vez configurado el espejo se edita el archivo ***/etc/vfstab***, que contiene la información de los sistemas de archivos que debe montar el sistema operativo al momento de arrancar. Se cambia el punto de montura de las particiones por el metadispositivo espejo creado.

```
uxsop001 / # cat /etc/vfstab
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
#
fd      -       /dev/fd fd      -       no      -
/proc   -       /proc   proc    -       no      -
/dev/md/dsk/d1  -       -       swap    -       no      -
/dev/md/dsk/d0  /dev/md/rdsk/d0 /       ufs     1       no      -
/dev/md/dsk/d4  /dev/md/rdsk/d4 /var/crash      ufs     2       yes     -
/devices        -       /devices        devfs   -       no      -
sharefs -       /etc/dfs/sharetab       sharefs -       no      -
ctfs    -       /system/contract        ctfs    -       no      -
objfs   -       /system/object  objfs   -       no      -
swap    -       /tmp    tmpfs   -       yes     -
```

Es necesario reiniciar el sistema operativo para que levante los nuevos metadispositivos espejo. Una vez validado, se procede a juntar los subespejos del disco espejo.

```
uxsop001 / # metattach d0 d20
uxsop001 / # metattach d1 d21
uxsop001 / # metattach d4 d24
```

Finalmente con el comando **metastat** se monitorea el proceso de sincronización del primer disco al disco espejo, quedando configurado el espejo para los discos de arranque (Figura 4.14).

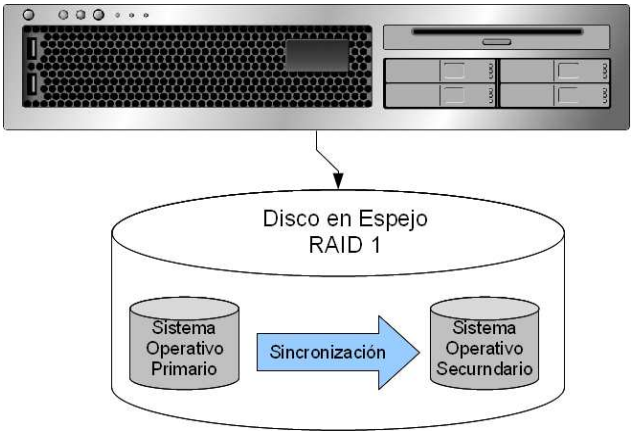



Figura 4.14 Creación del Espejo Final

## 4.3. Configuración de Solaris Containers

Este subtema abarca la configuración de los Solaris Containers, tema principal de este trabajo. En el desarrollo del subtema se muestra un resumen de la configuración de los Solaris Containers, el detalle de toda la configuración se encuentra en el Anexo 3.

La configuración de los Solaris Containers implica ciertos pasos, los cuales se desarrollan más adelante en este subtema y son:

- Configuración del Pool de Recursos
- Definición de Zonas

### 4.3.1. Configuración Pool de recursos

Las zonas ya sean globales o locales tienen asociado un *Resource Pool*, que es una entidad lógica que contiene recursos del sistema, como lo son CPU y memoria. Este Resource Pool puede ser independiente o estar de forma compartida con una o varias zonas.

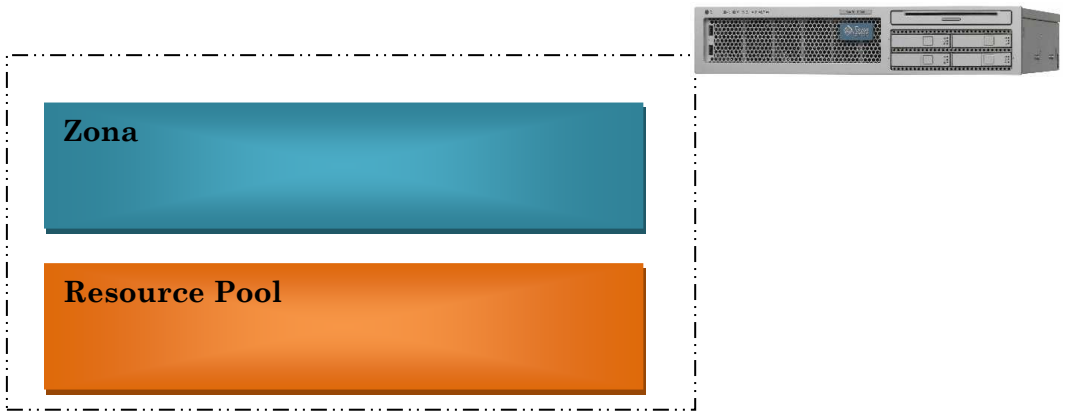



Figura 4.15 Configuración de Solaris Container

A continuación describimos brevemente la configuración y creación de los pool's de recursos para el Servidor Aplicativo de Colaboración.

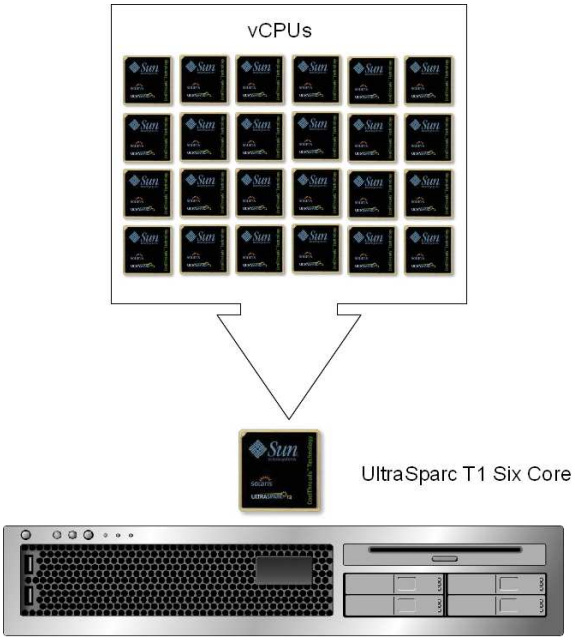



Figura 4.16 Pool de Recursos Inicial

El demonio poold es el encargado de controlar y proveer facilidades de particionamiento de recursos a la hora de crear los Pool's de Recursos. Para poder llevar a cabo la configuración de los Pool's de Recursos para las zonas que vamos a crear, es necesario verificar que antes de iniciar la configuración, el demonio poold esté activado, si no está activado se debe proceder a habilitarlo.

Verificamos que el demonio *poold* se encuentre activado mediante el comando **pgrep** y el parámetro –**lf poold,** posteriormente verificamos si los servicios se encuentran habilitados mediante el comando **svcs**.

```
uxsop001 / # pgrep -lf poold
uxsop001 / #
uxsop001 / # svcs -a |grep pools
disabled       Oct_23   svc:/system/pools/dynamic:default
online         Oct_24   svc:/system/pools:default
```

Se puede observar que el servicio de "Resource Pool" se encuentra habilitado, mientras que el servicio de "Resource Pool Dinámico" no lo está, ya que este servicio es dependiente del primero, para habilitarlo utilizamos el comando **svcadm** con los parámetros **enable** –**r** , especificando que habilitaremos los dos servicios:

```
uxsop001 / #
uxsop001 / # svcadm enable -r pools/dynamic
uxsop001 / #
```

Una vez habilitados los servicios verificamos su estado mediante el comando **svcs**. En la columna de estado observamos que ambos servicios se encuentran habilitados.

```
uxsop001 / # svcs -a |grep pools
online         13:30:18 svc:/system/pools/dynamic:default
online         Oct_24   svc:/system/pools:default
```

Nuevamente obtenemos el estado del demonio *poold* mediante **pgrep** y observamos que el demonio se encuentra activo.

```
uxsop001 / # pgrep -lf poold
28467 /usr/lib/pool/poold
```

Para la configuración de los Pool's, se usa el comando poolcfg, el cual provee todas las operaciones necesarias para esta tarea. Inicialmente utilizando el comando poolcfg con el argumento –c, creamos el primer conjunto de procesadores (*pset*) que en nuestro caso contendrá 2 vCPU como recurso de procesamiento para el contendor de Servidor de Sitio de Colaboración.

Después de crear el conjunto de procesadores, se deben de especificar como parámetros el número de procesadores virtuales (vCPU's) que va a tener el *pset* recién creado para el Servidor del Sitio de Colaboración y es por ello que especificamos que máximo tiene 2 y mínimo tiene 2.

También se le define el nombre, para nuestro caso el nombre es colaboración-pset. Posteriormente se crea el pool de recursos para este *pset* mediante el comando **poolcfg** y el argumento **-c**.

```
uxsop001 / #poolcfg -c 'create pset pset_collabapp'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pset pset_collabapp (unit pset.min = 2;
unit pset.max = 2)'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pool pool_collabapp'
```

Una vez creados el *pset* y el *pool*, se requiere crear un vínculo entre ambos. Para este proceso se utiliza **associate pool** y posteriormente se activa esta configuración mediante el comando **pooladm –c**.

```
uxsop001 / #poolcfg -c 'associate pool pool_collabapp (pset pset collabapp)'
uxsop001/ #
```

En este momento ya tenemos creado el pool de recursos para el Servidor del Sitio de Colaboración, solamente verificamos la existencia del nuevo pool utilizando el comando **poolcfg -dc info**.

```
uxsop001 / # poolcfg -dc info

system default
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        string  system.poold.objectives wt-load

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pool pool_collabapp
                int     pool.sys_id 8
                boolean pool.active true
                boolean pool.default false
                int     pool.importance 1
                string  pool.comment
                pset    pset_collabapp

        pset pset_collabapp
                int     pset.sys_id 1
                boolean pset.default false
                uint    pset.min 2
                uint    pset.max 2
                string  pset.units population
                uint    pset.load 0
                uint    pset.size 2
                string  pset.comment

                cpu
                        int     cpu.sys_id 17
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 16
                        string  cpu.comment
                        string  cpu.status on-line
```

```
                cpu
                        int     cpu.sys_id 17
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 16
                        string  cpu.comment
                        string  cpu.status on-line
```

Enseguida se configuran y crean los pool´s de recursos para los demás contenedores.

Creación del POOL de Recursos Collab DB

```
uxsop001 / #poolcfg -c 'create pset pset_collabdb'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pset pset_collabdb (unit pset.min = 2; unit
pset.max = 2)'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pool pool_collabdb'
uxsop001 / #
uxsop001 / #poolcfg -c 'associate pool pool_collabdb (pset pset_collabdb)'
```

Creación del POOL de Recursos Joomla App

```
uxsop001 / #poolcfg -c 'create pset pset_joomlaapp'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pset pset_joomlaapp (unit pset.min = 4; unit
pset.max = 4)'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pool pool_joomlaapp'
uxsop001 / #
uxsop001 / # poolcfg -c 'associate pool pool_joomlaapp (pset pset_joomlaapp)'
```

Creación del POOL de Recursos Joomla DB

```
uxsop001 / #poolcfg -c 'create pset pset_joomladb'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pset pset_joomladb (unit pset.min = 2; unit
pset.max = 2)'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pool pool_joomladb'
uxsop001 / #
uxsop001 / #poolcfg -c 'associate pool pool_joomladb (pset pset_joomladb)'
```

Creación del POOL de Recursos Ecommerce DB

```
uxsop001 / #poolcfg -c 'create pset pset_ecommercedb'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pset pset_ecommercedb (unit pset.min = 2; unit
pset.max = 2)'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pool pool_ecommercedb'
uxsop001 / #
uxsop001 / #poolcfg -c 'associate pool pool_ecommercedb (pset
pset_ecommercedb)'
```

Creación del POOL de Recursos Ecommerce App

```
uxsop001 / #poolcfg -c 'create pset pset_ecommerceapp'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pset pset_ecommerceapp (unit pset.min = 8;
unit pset.max = 8)'
uxsop001 / #
uxsop001 / #poolcfg -c 'create pool pool_ecommerceapp'
uxsop001 / #
uxsop001 / #poolcfg -c 'associate pool pool_ecommerceapp (pset
pset_ecommerceapp)'
```

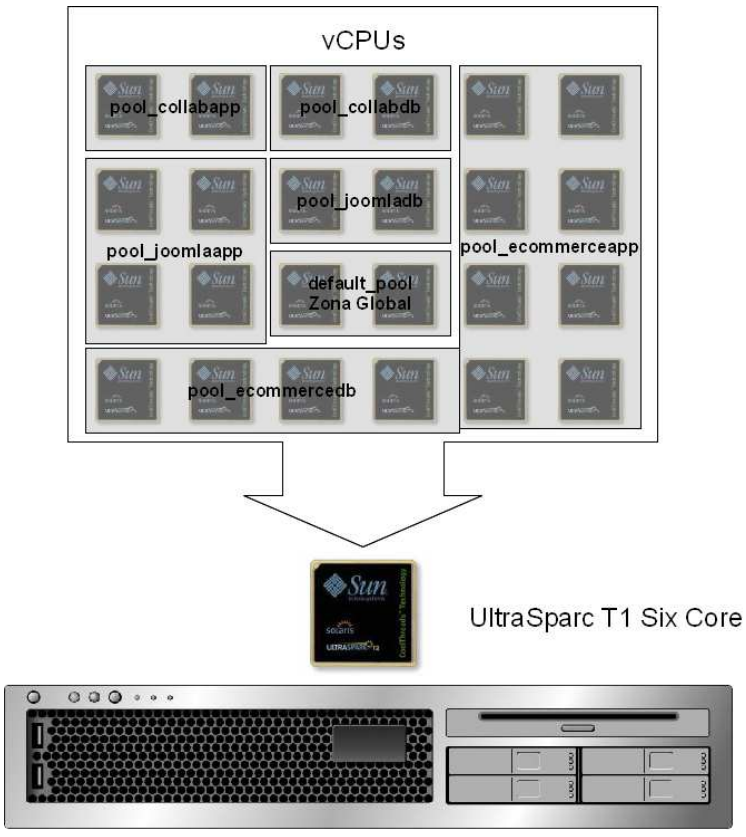



Figura 4.17 Pool de Recursos Final

## 4.4. Definición Zonas

Como se había mencionado en el capítulo 2, el sistema operativo base Solaris 10 contiene la Zona Global que se encarga de administrar el uso y acceso de los recursos, de la misma forma que permite crear, instalar y administrar las Zonas Locales o Zonas no Globales.

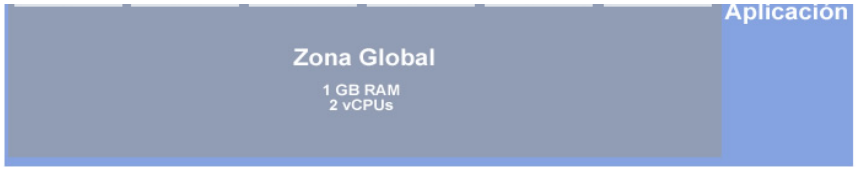



Figura 4. 18 Zona Global

En esta sección explicaremos brevemente la creación, instalación y administración de dichas Zonas Locales, las cuales son del tipo **whole root zone,** esto con la finalidad de proveer independencia a cada zona, ya que en las de tipo **sparse root zone** se instalan solamente ciertos directorios y hereda por default otros de solo lectura de la zona global como son /**lib**, /**platform**, /**sbin** y /**usr** y como consecuencia son dependientes de la zona global.

### Descripción de la creación de la Zona Joomla Aplicación

Las zonas locales tienen un filesystem del tipo ZFS, es por ello que en primera instancia creamos este sistema de archivos mediante el comando create, el cual requiere como parámetro el directorio del pool y el nombre de la zona, en nuestro caso zones y con el nombre de joomla-app respectivamente. Restringimos el tamaño que ocupará la zona en el disco mediante **set quota**, se le establece como tamaño 10 GB, espacio suficiente para la instalación y el buen desempeño de la aplicación.

```
uxsop001 / #zfs create zones/joomla-app
uxsop001 / #
uxsop001 / #zfs set quota=10g zones/joomla-app
```

Una vez creado el sistema de archivos utilizamos el comando **zonecfg** con el parámetro –**z** y definimos el nombre de la zona a crear, para nuestro caso **joomla-app.**

Este comando nos indica que no existe la zona configurada con la leyenda: "*No such zone configured*"; pero abre el Shell **zonecfg:joomla-app>.**

El Shell nos permite la creación de las zonas mediante el uso del comando **create** con el parámetro **–b,** para indicar que se trata de una nueva zona de tipo ***whole root zone*.**

```
uxsop001 / #zonecfg -z joomla-app
joomla-app: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:joomla-app> create -b
```

Una vez creada la zona la estableceremos en la ruta del directorio raíz con el comando **set zonepath**: El directorio de la zona local debe tener permisos del modo 755 (todos los privilegios para el usuario root, lectura y ejecución para otros y el grupo), ya que de esta forma se asegura que los usuarios sin privilegios de la zona global no pasen a un sistema de archivos de una zona local. Adicionalmente se debe especificar si la zona tendrá el parámetro de **autoboot activado,** en nuestro caso le configuramos que **true**, esto para que de forma automática reinicie.

```
uxsop001 / #zonecfg -z joomla-app
joomla-app: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:joomla-app> create –b
zonecfg:joomla-app>
zonecfg:joomla-app> set autoboot=true
zonecfg:joomla-app> set zonepath=/zones/joomla-app
```

Los parámetros de red los especificamos con el comando **add net**, una vez dentro del Shell definimos la dirección IP que se va a utilizar mediante **set address**, seguido de la IP **192.9.198.213.** Posteriormente le asignamos dicha IP a la interfaz de red **e1000g0** mediante **set physical** y finalizamos la asignación con **end**.

```
uxsop001 / #zonecfg -z joomla-app
joomla-app: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:joomla-app> create –b
zonecfg:joomla-app>
zonecfg:joomla-app> set autoboot=true
zonecfg:joomla-app> set zonepath=/zones/joomla-app
zonecfg:joomla-app>
zonecfg:joomla-app> add net
zonecfg:joomla-app:net>
zonecfg:joomla-app:net> set address=192.9.198.213
zonecfg:joomla-app:net> set physical=e1000g0
zonecfg:joomla-app:net> end
```

El comando **verify** se utiliza para verificar si sintácticamente es correcta la configuración, podemos observar que al no obtener ningún error la zona.

Finalmente con el comando **commit,** se mueve la configuración de la memoria a disco. Para finalizar el proceso tecleamos **exit**.

```
uxsop001 / #zonecfg -z joomla-app
joomla-app: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:joomla-app> create –b
zonecfg:joomla-app>
zonecfg:joomla-app> set autoboot=true
zonecfg:joomla-app> set zonepath=/zones/joomla-app
zonecfg:joomla-app>
zonecfg:joomla-app> add net
zonecfg:joomla-app:net>
zonecfg:joomla-app:net> set address=192.9.198.213
zonecfg:joomla-app:net> set physical=e1000g0
zonecfg:joomla-app:net> end
zonecfg:joomla-app>
zonecfg:joomla-app> verify
zonecfg:joomla-app> commit
zonecfg:joomla-app> exit
```

La instalación se lleva a cabo a través del comando **zoneadm** con la opción **z** seguido del nombre de la zona y después **install**, el cual verifica la disponibilidad de los recursos y además crea los puntos de montajes e instala la zona. Durante este proceso se muestran mensajes del avance los archivos y directorios instalados, al igual que informa errores en caso de que estos ocurran.

```
uxsop001 / # zoneadm -z colaboracion-app install
cannot create ZFS dataset zones/colaboracion-app: dataset already exists
Preparing to install zone <colaboracion-app>.
Creating list of files to copy from the global zone.
Copying <148832> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <1205> packages on the zone.
Initialized <1205> packages on zone.
Zone <colaboracion-app> is initialized.
Installation of <4> packages was skipped.
Installation of these packages generated warnings: <SUNWtcatr>
The file </zones/colaboracion-app/root/var/sadm/system/logs/install_log> contains a log of the zone
 installation.
uxsop001 / #
```

La primera zona fue creada e instalada con éxito, no se obtuvo ningún error por lo procedemos a la creación y configuramos el resto de las zonas.

Configuración y Creación de la zona Joomla-DB

```
uxsop001 / #zfs create zones/joomla-db
uxsop001 / #
uxsop001 / #zfs set quota=10g zones/joomla-db
uxsop001 / #
uxsop001 / #zonecfg -z joomla-db
joomla-db: No such zone configured
Use 'create' to begin configuring a new zone.
```

```
uxsop001 / #zfs create zones/joomla-db
uxsop001 / #
uxsop001 / #zfs set quota=10g zones/joomla-db
uxsop001 / #
uxsop001 / #zonecfg -z joomla-db
joomla-db: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:joomla-db> create -b
zonecfg:joomla-db>
zonecfg:joomla-db> set autoboot=true
zonecfg:joomla-db> set zonepath=/zones/joomla-db
zonecfg:joomla-db>
zonecfg:joomla-db> add net
zonecfg:joomla-db:net>
zonecfg:joomla-db:net> set address=192.9.198.212
zonecfg:joomla-db:net> set physical=e1000g0
zonecfg:joomla-db:net> end
zonecfg:joomla-db>
zonecfg:joomla-db> verify
zonecfg:joomla-db> commit
zonecfg:joomla-db> exit
```

Configuración y Creación de la zona Colaboración-APP

```
uxsop001 / #zfs create zones/colaboracion-app
uxsop001 / #
uxsop001 / #zfs set quota=10g zones/colaboracion-app
uxsop001 / #
uxsop001 / #zonecfg -z colaboracion-app
colaboracion-app: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:colaboracion-app> create -b
zonecfg:colaboracion-app>
zonecfg:colaboracion-app> set autoboot=true
zonecfg:colaboracion-app> set zonepath=/zones/colaboracion-app
zonecfg:colaboracion-app>
zonecfg:colaboracion-app> add net
zonecfg:colaboracion-app:net>
zonecfg:colaboracion-app:net> set address=192.9.198.185
zonecfg:colaboracion-app:net> set physical=e1000g0
zonecfg:colaboracion-app:net> end
zonecfg:colaboracion-app>
zonecfg:colaboracion-app> verify
zonecfg:colaboracion-app> commit
zonecfg:colaboracion-app> exit
```

Configuración y Creación de la zona Colaboración-DB

```
uxsop001 / #zfs create zones/colaboracion-db
uxsop001 / #
uxsop001 / #zfs set quota=10g zones/colaboracion-db
```

```
uxsop001 / #zonecfg -z colaboracion-db
colaboracion-db: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:colaboracion-db> create -b
zonecfg:colaboracion-db>
zonecfg:colaboracion-db> set autoboot=true
zonecfg:colaboracion-db> set zonepath=/zones/colaboracion-db
zonecfg:colaboracion-db>
zonecfg:colaboracion-db> add net
zonecfg:colaboracion-db:net>
zonecfg:colaboracion-db:net> set address=192.9.198.91
zonecfg:colaboracion-db:net> set physical=e1000g0
zonecfg:colaboracion-db:net> end
zonecfg:colaboracion-db>
zonecfg:colaboracion-db> verify
zonecfg:colaboracion-db> commit
zonecfg:colaboracion-db> exit
```

Configuración y Creación de la zona Ecommerce-DB

```
uxsop001 / #zfs create zones/ecommerce-db
uxsop001 / #
uxsop001 / #zfs set quota=10g zones/ecommerce-db
uxsop001 / #
uxsop001 / #zonecfg -z ecommerce-db
ecommerce-db: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:ecommerce-db> create -b
zonecfg:ecommerce-db>
zonecfg:ecommerce-db> set autoboot=true
zonecfg:ecommerce-db> set zonepath=/zones/ecommerce-db
zonecfg:ecommerce-db>
zonecfg:ecommerce-db> add net
zonecfg:ecommerce-db:net>
zonecfg:ecommerce-db:net> set address=192.9.198.71
zonecfg:ecommerce-db:net> set physical=e1000g0
zonecfg:ecommerce-db:net> end
zonecfg:ecommerce-db>
zonecfg:ecommerce-db> verify
zonecfg:ecommerce-db> commit
zonecfg:ecommerce-db> exit
```

Configuración y Creación de la zona Ecommerce-App

```
uxsop001 / #zfs create zones/ecommerce-app
uxsop001 / #
uxsop001 / #zfs set quota=10g zones/ecommerce-app
uxsop001 / #
uxsop001 / #zonecfg -z ecommerce-app
ecommerce-app: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:ecommerce-app> create -b
zonecfg:ecommerce-app>
zonecfg:ecommerce-app> set autoboot=true
zonecfg:ecommerce-app> set zonepath=/zones/ecommerce-app
zonecfg:ecommerce-app>
zonecfg:ecommerce-app> add net
zonecfg:ecommerce-app:net>
zonecfg:ecommerce-app:net> set address=192.9.198.235
zonecfg:ecommerce-app:net> set physical=e1000g0
zonecfg:ecommerce-app:net> end
zonecfg:ecommerce-app>
zonecfg:ecommerce-app> verify
zonecfg:ecommerce-app> commit
zonecfg:ecommerce-app> exit
```

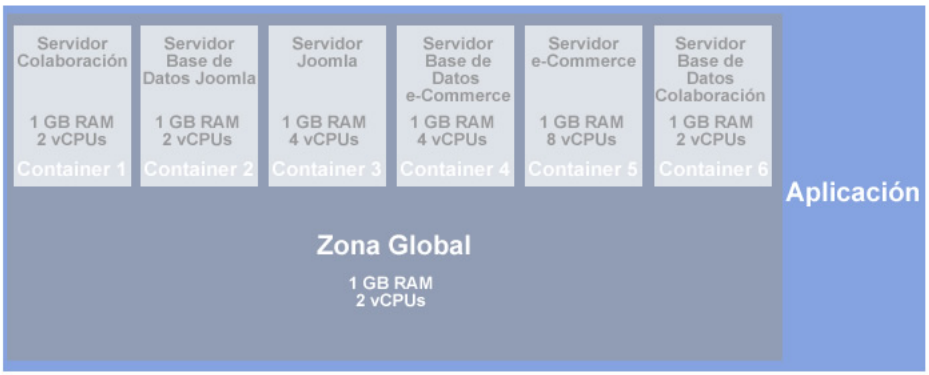



Figura 4.18 Configuración Final Zona Global y zonas locales

Ya que se tienen creadas las zonas (Figura 4.18), procedemos a la instalación de cada una de ellas. Mediante el comando **zoneadm list** y el parámetro –**cv** obtenemos la información de las zonas, la primera columna indica el "*ID Name"* que es el nombre de la zona, la siguiente columna indica el estado, aquí podemos observar que todas las zonas se encuentran instaladas y que tienen una dirección IP del tipo *Shared*.

```
uxsop001 / # zoneadm list -cv
  ID NAME             STATUS     PATH                           BRAND    IP
   0 global           running    /                              native   shared
   9 colaboracion-app installed  /zones/colaboracion-app        native   shared
  10 colaboracion-db  installed  /zones/colaboracion-db         native   shared
  11 ecommerce-db     installed  /zones/ecommerce-db            native   shared
  12 ecommerce-app    installed  /zones/ecommerce-app           native   shared
  13 joomla-db        installed  /zones/joomla-db               native   shared
  14 joomla-app       installed  /zones/joomla-app              native   shared
```

Después de realizar este procedimiento, continuamos a iniciar sesión en cada zona con la finalidad de realizar la instalación de Solaris, la cual se realiza de forma muy similar a la instalación del Sistema Operativo base y con los mismos parámetros de configuración. Se inicia sesión con el comando **zlogin –C** y el nombre de la zona. Posteriormente seleccionamos el idioma, en nuestro caso inglés que es la opción **0** (Figura 4.19), adicionalmente especificamos en **locale** la opción **0** que es la adecuada para el teclado, ya que la otras opciones son para sitios geográficos en Canadá y USA:

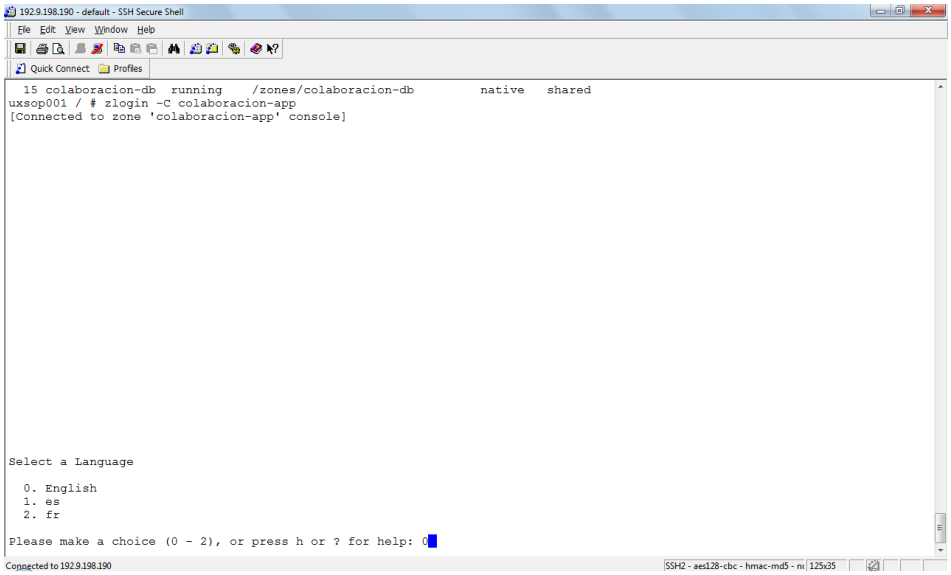



Figura 4.19 Conexión a consola de la zona colaboración-app

En la siguiente pantalla seleccionamos el tipo d terminal que estamos usando, en nuestra instalación se utilizó la opción **3**, correspondiente a DEC VT100.

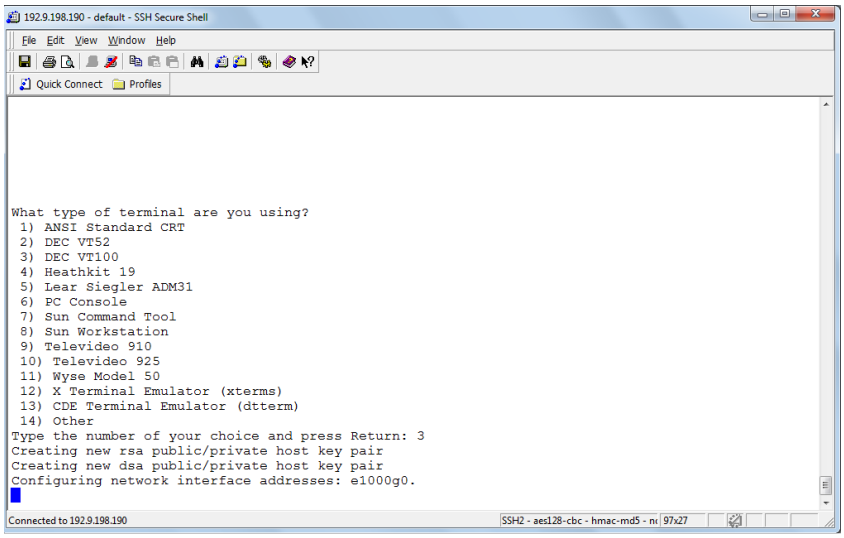



Figura 4.20 Configuración inicial de la zona colaboración-app

Una vez hecho esto especificamos los parámetros de la configuración como son el nombre del host, políticas de seguridad de kerberos, servicios habilitados, etc., las configuraciones para cada contenedor se detallan en el Anexo 3 de este documento.

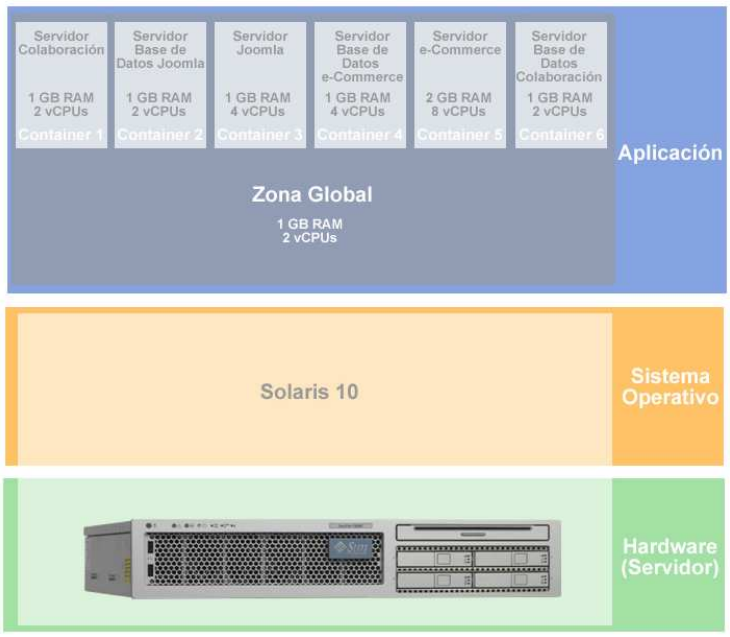
**Asignación de Memoria RAM**

Para la asignación de memoria RAM para las zonas creadas, decidimos usar la técnica llamada *"Memory Capping",* la cual permite establecer un límite máximo a los recursos de memoria física. Tanto en Zonas como en Proyectos de Solaris 10, la definición de límites de memoria física es posible.

Para este trabajo de Tesis, decidimos usar la técnica de "*Memory Capping"* ya que de esta manera se establece un control administrado del uso de memoria por cada Zona, es decir, cada Zona puede trabajar con el máximo de memoria física establecida sin agotar los recursos de memoria de otras zonas. Cuando se decide usar la técnica *"Memory Capping"*, se debe de tener muy clara la cantidad de RAM que necesita cada aplicativo que albergará una Zona, ya que si se establece un límite y éste no satisface las necesidades del aplicativo, se tendrán problemas de rendimiento y el aplicativo no funcionará.

El demonio *rcapd* es el encargado de proporcionar los mecanismos necesarios para la asignación y administración de los límites de memoria ó *"Memory Capping"*.

La asignación de memoria por Zona, se definió en el capítulo anterior y se muestra en la figura 4.21:




4.21 Detalle de Memoria Física por Contenedor

A continuación se describe brevemente la configuración de *"Memory Capping"* para la Zona *colaboracion-db*:

Primero, se tiene que verificar que el servicio *rcap* se encuentre activo, de otra manera no se podrá realizar la configuración. Con el comando **svcadm** y el argumento **enable** habilitamos el servicio *rcap*:

```
uxsop001# svcadm enable rcap
```

Para ejecutar estos comandos, se debe de contar con privilegios de super usuario del sistema. Para que el demonio *rcapd* inicie inmediatamente y también inicialice cada vez que el sistema inicia, se usa el comando **rcapadm** con el argumento **–E:**

```
uxsop001# rcapadm –E
```

Una vez que habilitamos el servicio *rcap*, podemos comenzar con la configuración de los límites de memoria para las zonas.

El detalle de configuración de memoria para todas las zonas, se muestra en el Anexo 4.

A continuación se muestra la configuración para la zona colaboracion-db.

```
uxsop001# zonecfg –z colaboración-db
zonecfg:colaboracion-db> add capped-memory
zonecfg:colaboración-db:capped-memory> set physical=1g
zonecfg:colaboracion-db:capped-memory> end
zonecfg:colaboracion-db> verify
zonecfg:colaboración-db> commit
zonecfg:colaboración-db> exit
```

Para finalizar la configuración, debe de reiniciarse la zona:

```
uxsop001# zlogin colaboración-db init 6
```

El mismo procedimiento debe de realizarse con el resto de las Zonas.

### 4.5. Instalación de Webstack

Posterior a la instalación y configuración del Sistema Operativo Base, la generación y configuración de los Contenedores, es necesario instalar el Software que servirá para dar servicio, tal como se ha planteado en la sección de Diseño del presente documento.

Para este trabajo de Tesis se ha usado un producto que cumple con los requerimientos de los aplicativos descritos, llamado Webstack, producto de la empresa Sun Microsystems, que integra un servidor web Apache, un intérprete de PHP compatible y pre-configurado para funcionar con Apache; y por último MySQL. Esta combinación es conocida como un Stack AMP (Apache MySQL PHP), así mismo es muy común su uso para aplicaciones web. La ventaja de usar Sun Webstack, en lugar de otros productos que integran los mismos elementos de software, es que fue desarrollado por la misma empresa que el sistema operativo, con ello se reducen riesgos por compatibilidad así mismo tiene una mejor integración.

### 4.5.1. Servidor de Aplicaciones

Para el caso de un servidor basado en el Stack AMP, un Servidor de aplicaciones está compuesto por el servidor Web Apache y el intérprete de PHP. El primero es el que recibe las peticiones desde los clientes remotos, pasándolas al intérprete de PHP, quién, dependiendo de cómo haya sido concebida la aplicación, puede generar contenido dinámico o interactuar con la base de datos, regresando los datos procesados al servidor Web para ser mostrados al usuario.

La instalación de Apache y de PHP se realiza desde los archivos de instalación de Sun Webstack, que al ser integrado por el mismo Sun Microsystems reduce la operación a un par de comandos, uno instala el software y el otro lo activa.

La instalación de cualquier software en los Solaris Containers requiere que se esté dentro del Shell del contenedor, esto se logra con el comando **zlogin** y el nombre de la Zona desde la zona global (o sistema operativo base), el procedimiento se muestra en la figura 4.22.

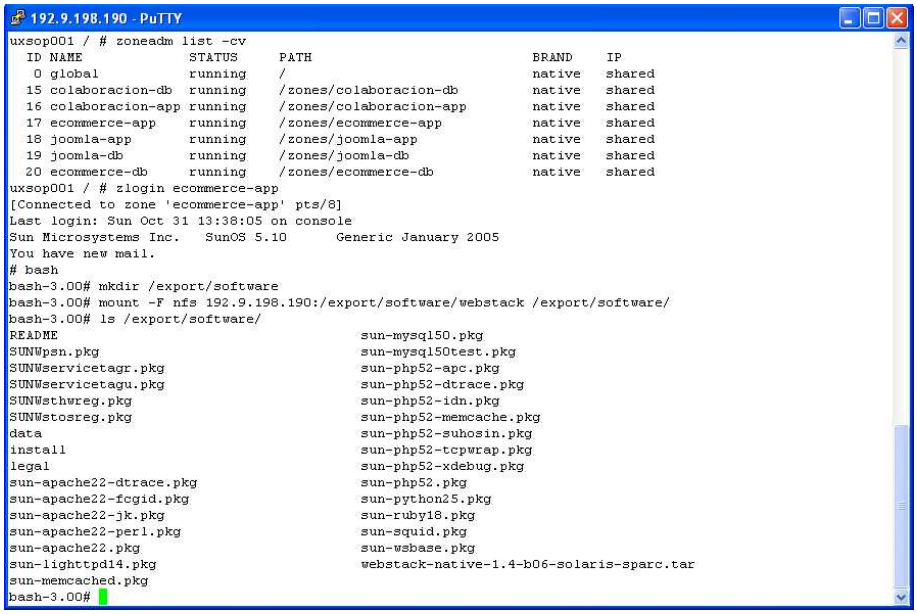



Figura 4.22 Conexión a la zona ecommerce-app

En la figura 4.23 se muestra la instalación de Apache y PHP mediante el instalador de Webstack, se asume que los archivos de instalación se encuentran en el Contenedor correspondiente. Desde el directorio donde se encuentran los archivos de instalación se ingresa el comando **./install php** y posteriormente **./install apache**, esta acción copia los archivos ejecutables o binarios, asimismo inicializa los archivos de configuración del Sistema Operativo.

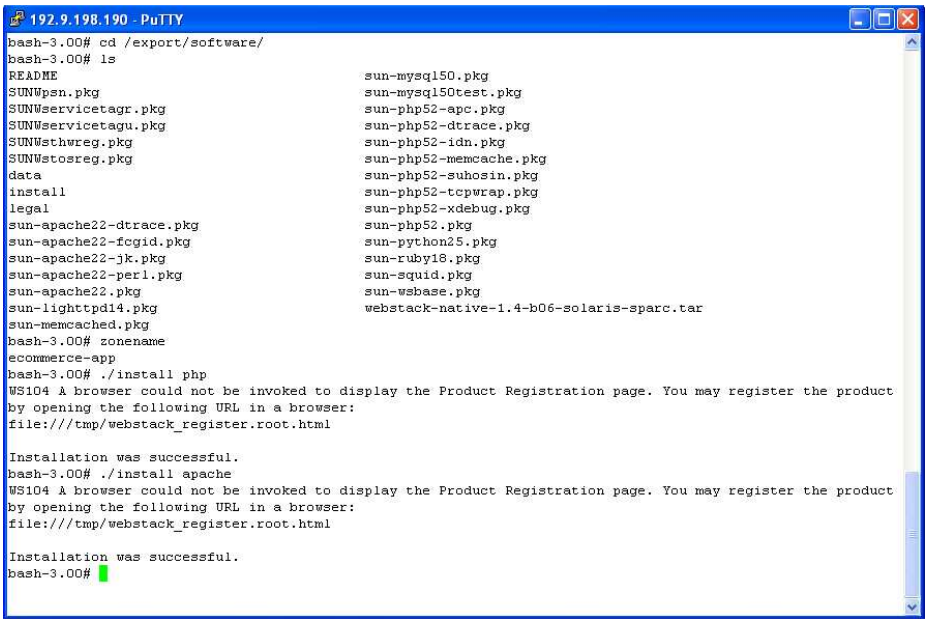



Figura 4.23 Instalación de Apache y PHP

Posterior a la instalación, es necesario inicializar el servicio, debido a la integración de Webstack con Solaris, esto se limita al comando **svcadm enable sun-apache22**, de otra forma, la secuencia de comandos depende del sistema operativo y de cada una de las compilaciones de Apache y PHP. Para inicializar y validar que Apache está activo, se utiliza el comando **svcs | grep apache**. Adicional a la activación, es necesario verificar la dirección IP correspondiente a la Zona (Figura 4.24).

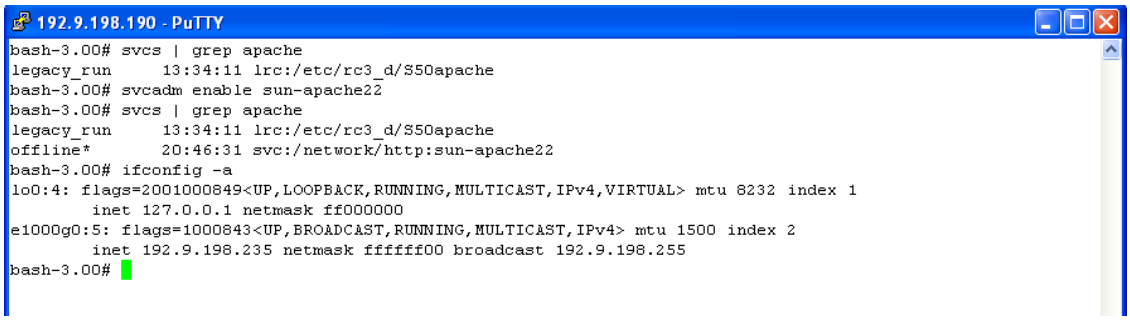



Figura 4.24 Configuración de red de la zona

Para validar que el servicio del servidor de aplicativos y web está activo, Webstack instala datos de ejemplo, los cuales pueden ser visualizados desde un navegador web, tecleando como dirección la dirección IP (Figura 4.25).

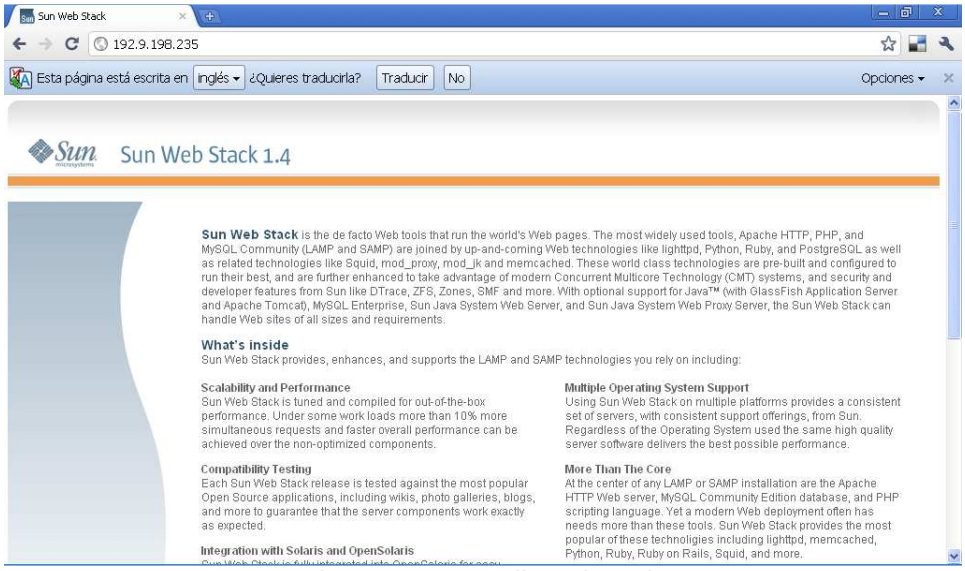



Figura 4.25 Pantalla webstack

### 4.5.2. Servidor de Base de datos

Un servidor de Base de Datos, es un sistema informático cuya función es la de ser un repositorio de datos, en particular son datos que residen en estructuras llamadas tablas.

Estos servidores son capaces de manipular y administrar dichas estructuras y la información que en ellas residen, además administran el acceso a la información.

Para este trabajo se usó el motor de base de datos de MySQL, debido a que forma parte de Webstack, por ello tiene una mejor integración con el entorno completo. Además MySQL es la base de datos en la que fueron desarrollados los aplicativos que estarán albergados en el servidor del trabajo de Tesis(Joomla, E-Commerce y Collabtive), cumple con la ventaja de ser de Código Abierto y de distribución libre, sin límites en la cantidad de bases de datos que pueda manipular.

Sin embargo, en la industria de las Tecnologías de la Información, existen otros manejadores de base de datos que también son de Código Abierto y de distribución libre, como es el caso de PostgreSQL, JavaDB y Apache Derby, de la misma forma existen manejadores de base de datos como Oracle y DB2 que cuentan con versiones de distribución libre, pero no son de código abierto, estas versiones están limitadas en capacidad o escalabilidad.

Cualquiera de estos manejadores de bases de datos tienen a SQL, Structured Query Languaje o Lenguaje de Consulta Estructurado, como lenguaje compatible entre ellos, sin embargo existen sentencias propias de cada manejador y el cambiar entre uno y otro puede agregar un grado de dificultad considerable.

De la misma forma en que se instaló Apache y PHP en los Servidores de Aplicativos, la instalación de MySQL mediante el instalador de Webstack se realiza de manera sencilla, mediante los comandos **./install amp**, el cual hace la copia de los archivos ejecutables o binarios y configura los servicios de MySQL. Posterior a la instalación, es necesario inicializar el servicio de la misma forma en que se hizo con el Servidor de Aplicativos, mediante el comando **svcadm enable sun-mysql50** y el comando **svcs | grep mysql** para verificar que el servicio está inicializado (Figura 4.26).

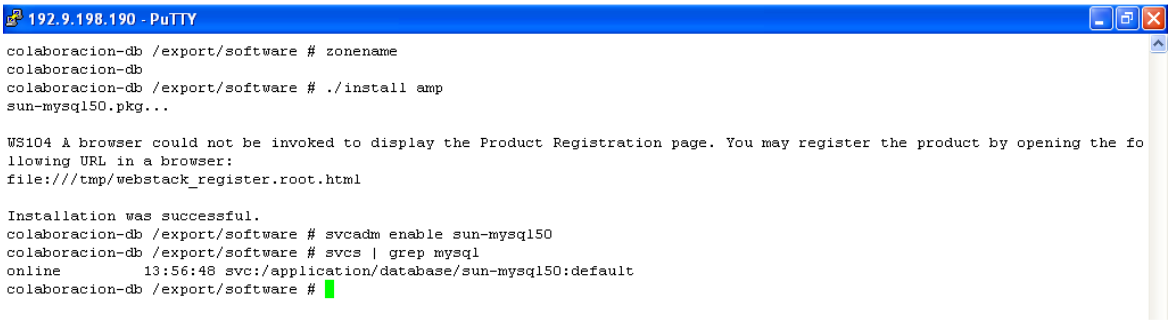



Figura 4.26 Habilitar el servicio de MySQL

La instalación del Webstack en cada uno de los Contenedores se realiza de la misma forma, dependiendo del propósito del Contenedor es el software que se instala, Apache y PHP en los servidores Web y de Aplicación y MySQL en los servidores de Base de Datos. Al finalizar la instalación en todos los contenedores, los elementos instalados se muestran en la figura 4.27.

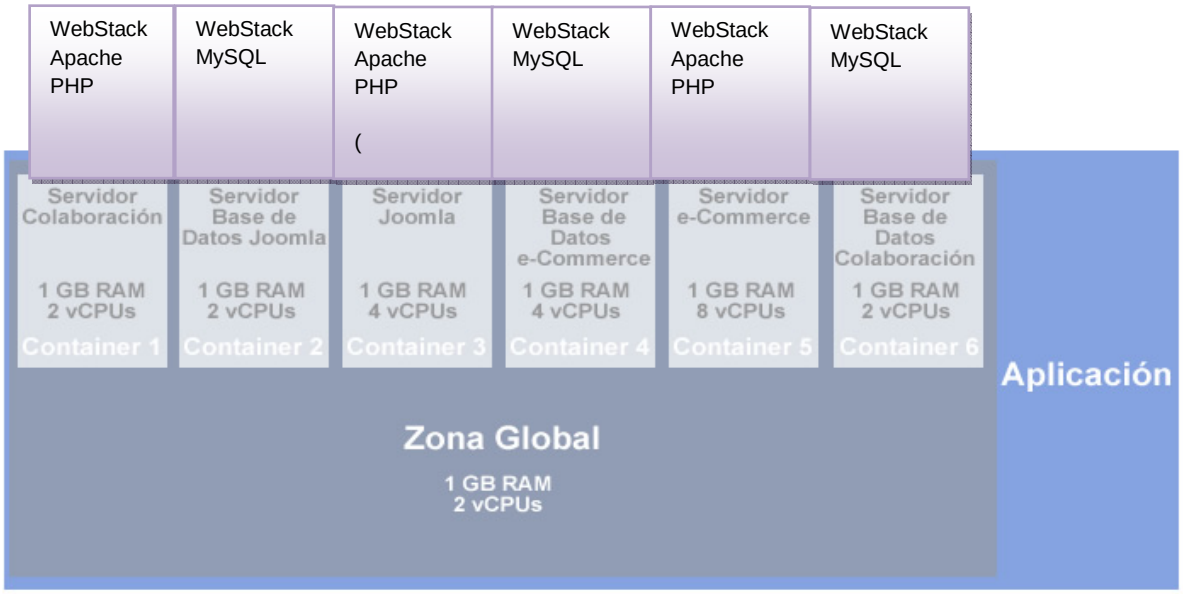



Figura 4.27 Instalación de Aplicaciones en los Solaris Containers

### 4.5.3. Configuración Webstack

Webstack fue configurado de acuerdo a cada una de las aplicaciones que en él residen, estas configuraciones incluyen desde creación de usuarios, inicialización de bases de datos y copia de archivos de las aplicaciones, culmina con la instalación y configuración de las aplicaciones.

El orden de configuración, preferentemente es Servidor de Base de Datos primero y luego el Servidor de Aplicaciones, esto debido a que es necesario tener un acceso y base de datos lista para instalar los aplicativos.

A continuación se muestra la configuración del Servidor de Base de Datos, en el proceso se revisan parámetros de red, estado del manejador de base de datos y posteriormente una configuración dentro del manejador de base de datos, la cual consiste en generar un usuario y su acceso a una base de datos específica.

En el Contenedor que corresponda al Servidor de base de datos, es necesario accesar a la consola de MySQL (Figura 4.28).

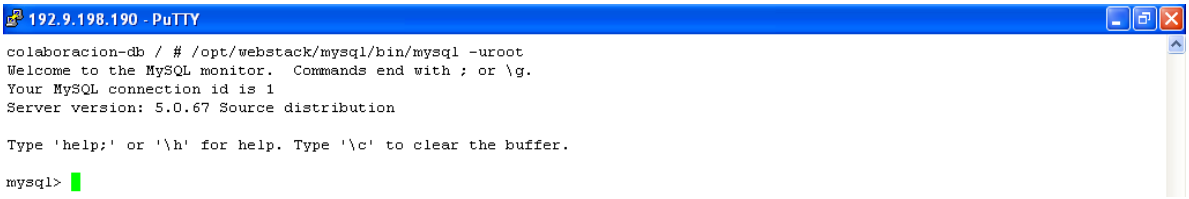



Figura 4.28 Conexión al servidor de base de datos

Posteriormente se genera el usuario para la base de datos, los permisos de acceso y finalmente la base de datos es creada. A manera de estandarizar, la base de datos tiene el mismo nombre que el usuario creado, otorgándole todos los permisos de modificar toda la base de datos (Figura 4.29).

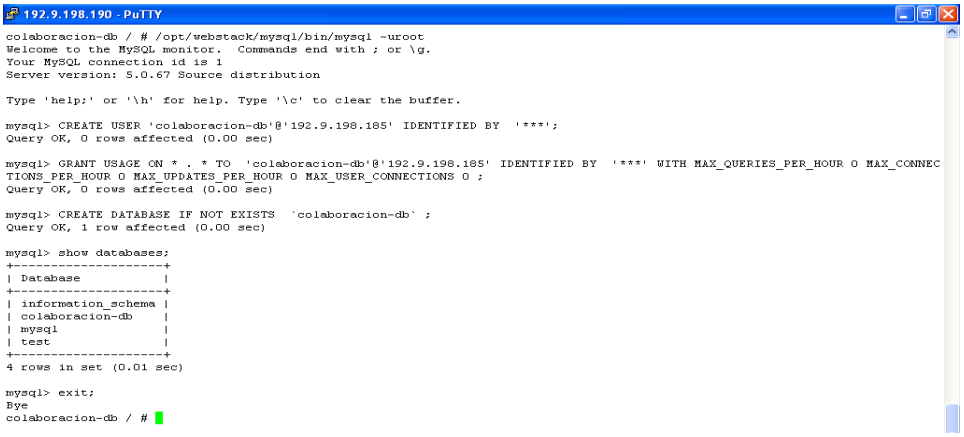



Figura 4.29 Configuración del servidor de base de datos

Con esto se finaliza la configuración del Servidor de Base de Datos. Seguido de este paso, es necesario configurar el aplicativo.

Primero se copian los archivos fuentes del aplicativo al directorio de http en Apache, el cual tiene una dirección específica en Webstack: */var/opt/webstack/apache2/2.2/htdocs/,* todo lo que se encuentre dentro de este directorio será publicado mediante Apache Web Server en respuesta a las peticiones que se hagan desde un navegador de internet. La figura 4.30 muestra como se hace la copia de los archivos desde un cliente de transferencias de archivos llamado FileZilla.

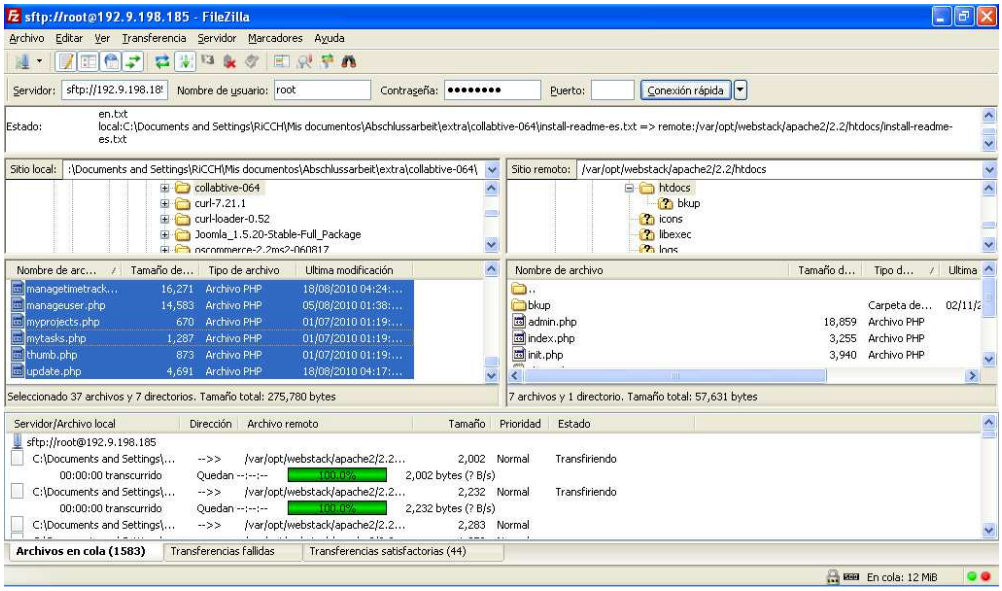



Figura 4.30 Transferencia de archivos

Es necesario hacer una verificación de los archivos y su integridad directamente en el Contenedor.

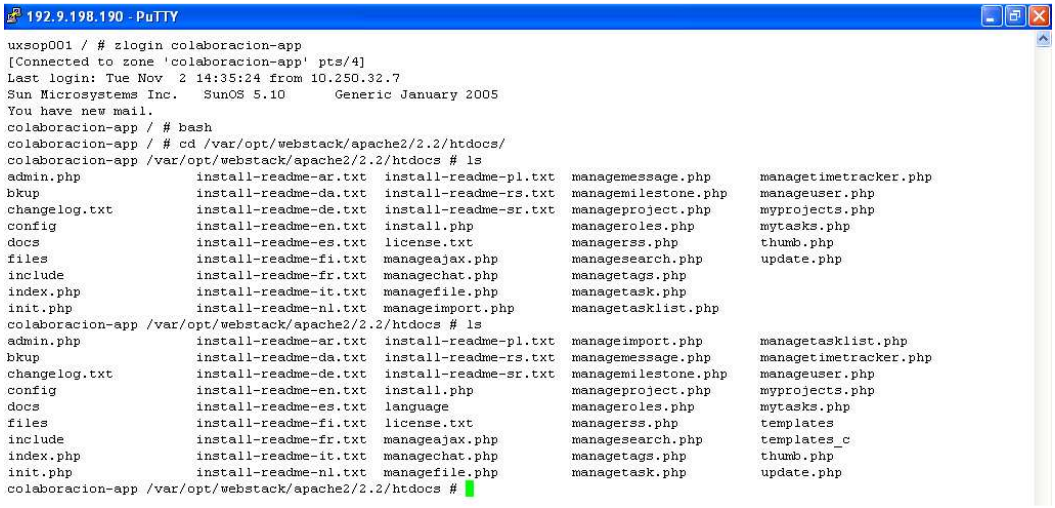



Figura 4.31 Integridad de archivos transferidos

En el caso particular del Servidor del Sitio de Colaboración, al momento de accesar mediante un navegador de internet, la siguiente página es la que se muestra:

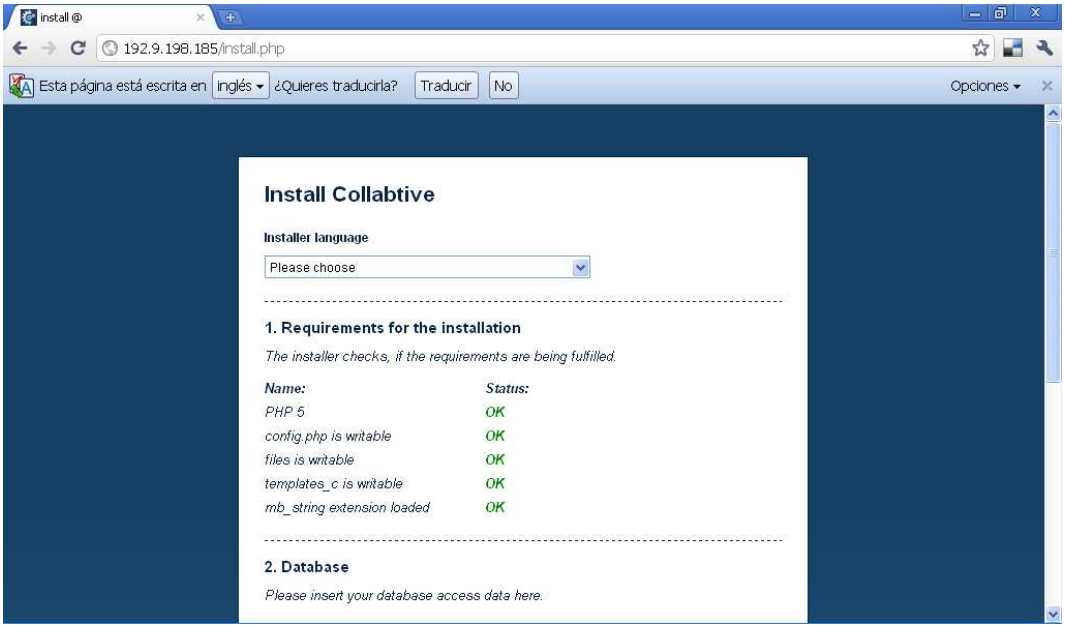



Figura 4.32 Pagina del Sitio de Colaboración

Esto nos indica que el servidor de aplicaciones está ejecutándose y respondiendo de manera adecuada. El proceso de instalación del aplicativo es único en cada caso, pero tiene puntos en común, la configuración de la base de datos requiere los parámetros de dirección IP, nombre de usuario y contraseña, tal cual se hizo en la configuración del servidor de base de datos.

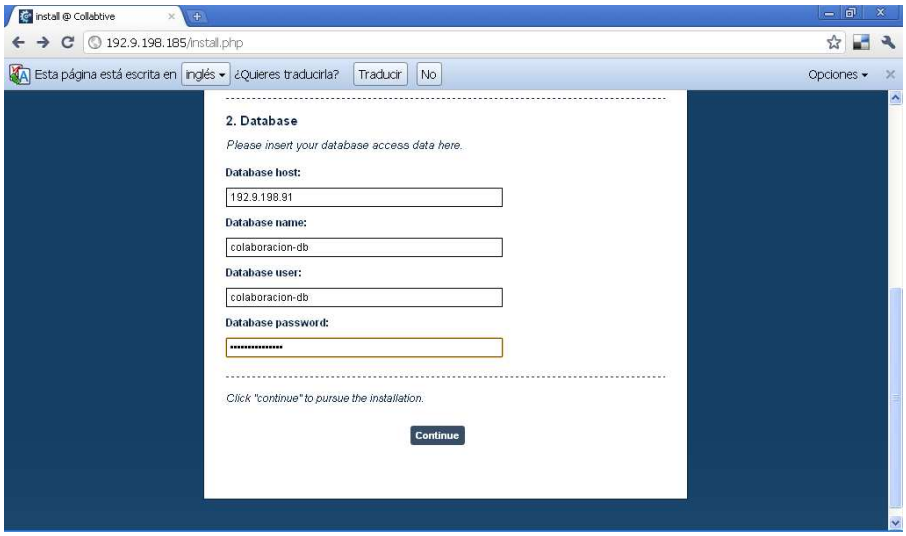



Figura 4.33 Configuración de la base de datos