Práctica MPI sobre VirtualBox

En este documento encontraréis toda la información necesaria para realizar la práctica de MPI utilizando la máquina virtual de VirtualBox.

Carga de la máquina virtual en Virtual Box

Para poder llevar a cabo la práctica se proporciona una máquina virtual (Alpine Linux 3.14.2 64 bits) preparada para VirtualBox. A continuación se explican los pasos para poder utilizarla.

Instalar VirtualBox

Si no tenéis VirtualBox instalado lo podéis descargar en la siguiente dirección:
https://www.virtualbox.org/wiki/Downloads.
Está disponible para Linux, Mac OS X y Windows, y nos permitirá ejecutar la máquina virtual Alpine Linux proporcionada, juntamente con este enunciado en cualquiera de los entornos anteriores.
Una vez VirtualBox esté instalado también tenemos que instalar "Oracle VM VirtualBox Extension Pack", que deberá ser exactamente la misma versión que tenemos de VirtualBox. Si tenemos una versión antigua deberemos buscarlo en la sección de "older builds".

Descargarse la Máquina Virtual

A continuación debéis descargaros la máquina virtual. Lo podéis hacer en la siguiente dirección:

https://drive.google.com/file/d/1egXbx8LRnzWpQ8_E8oZIPGcZW2jayrlz/view?usp=sharing

Para poder descargar el link es necesario tener sesión iniciada en drive.google.com con el usuario del campus de la UOC.

Instalar la Máquina Virtual

Una vez descargado el fichero ".ova" con la máquina virtual se debe importar a VirtualBox. Lo haremos mediante la opción Fichero / Importar Servicio Virtualizado (o también mediante doble clic directamente sobre el fichero).

Arrancar la Máquina Virtual

Una vez importado y si todo ha ido bien, cuando arranquemos la máquina nos aparecerá una pantalla solicitando usuario y password.
El usuario y el password de la máquina virtual son: root / student.

Para parar la máquina (una vez habéis hecho login y estáis dentro) podéis utilizar el siguiente comando:

```
sd-GCDA:~# poweroff
```

Cómo acceder a la máquina Virtual mediante ssh y sftp

Para trabajar con la máquina virtual es más práctico poder conectarnos desde el sistema operativo host (o anfitrión) utilizando *ssh*. Para poder hacerlo la máquina virtual viene preconfigurada con NAT de forma que el puerto 2522 del host redirige al puerto 22 del guest (máquina virtual).

Para acceder utilizamos cualquier herramienta que permita conectar por *ssh*. Por ejemplo en Linux esto se hace con el comando *ssh*:

```
ssh root@localhost
root@localhost's password:
Welcome to Alpine!
sd-GCDA:~#
```

En Windows herramientas habituales para trabajar por *ssh* son:
• Putty (http://www.putty.org/)
Cliente *ssh* muy usado en el mundo Windows.
• Filezilla (https://filezilla-project.org/)
Cliente *sftp* que nos permitirá transferir ficheros a la máquina virtual por *sftp*. Deberemos indicar los siguientes parámetros en la barra de conexión rápida:

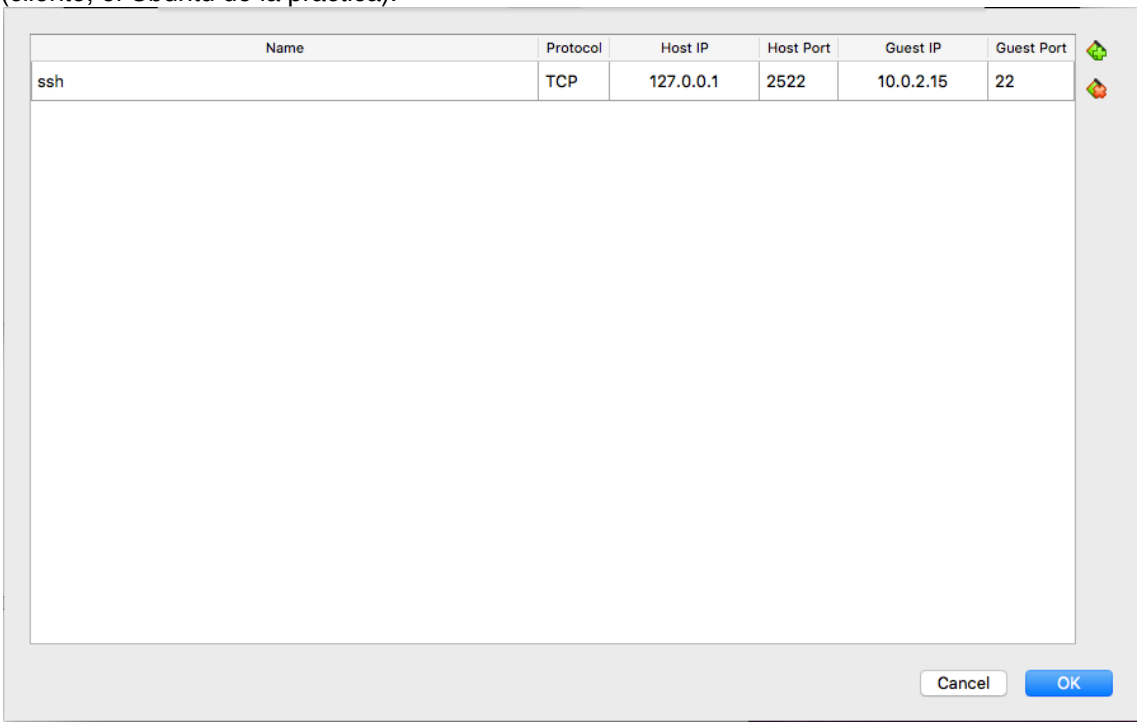
• Servidor: sftp://localhost
• Usuario: root
• Contraseña: student
• Puerto: 2522

Solución de problemas

Este apartado sólo es relevante si ha fallado alguno de los puntos anteriores y no habéis podido acceder correctamente a la máquina. Si habéis podido acceder correctamente lo podéis saltar.

NAT (Redirección de puertos)

La máquina virtual proporcionada ya tiene las redirecciones de puerto del host al guest necesarias. En caso de no poder acceder por *ssh* como se ha indicado en los apartados anteriores revisar que el NAT está bien configurado.
Nos situamos sobre la máquina virtual y en el apartado de red de los parámetros escogemos “Reenvío de puertos”. En esta pantalla debería indicar (sino, tenemos que añadirlo) que el puerto 2522 del host (anfitrión, es decir, nuestra máquina física) se redirige al puerto 22 del guest (cliente, el Ubuntu de la práctica).

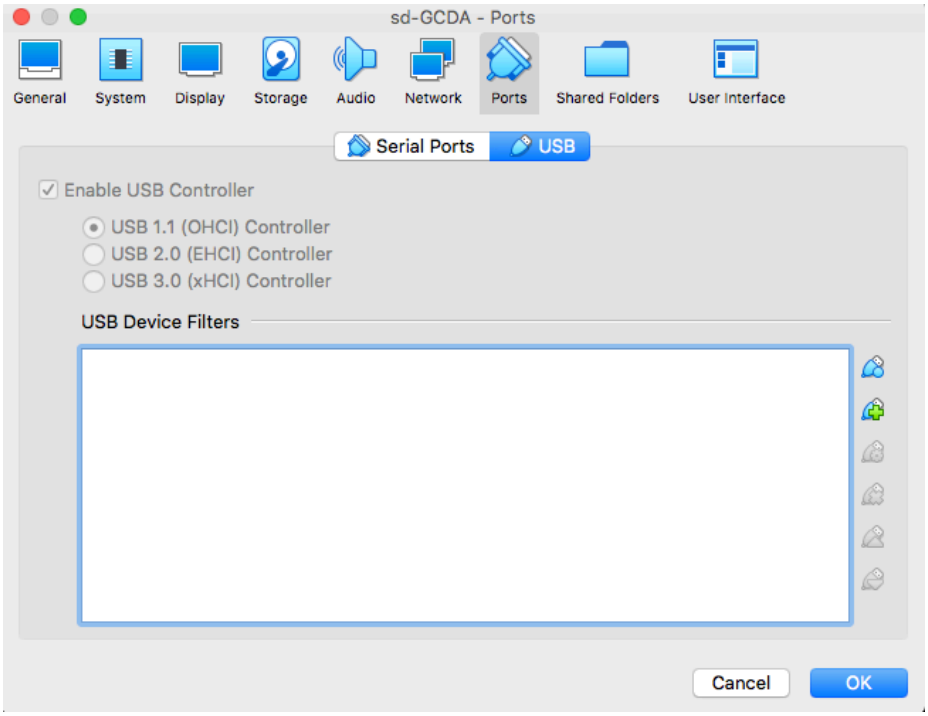


Asimismo, debemos validar que la IP del cliente es 10.0.2.15 mediante *ifconfig*. Y en caso de no serlo poner la IP asignada en nuestro caso:

```
sd-GCDA:~# ifconfig
eth0 Link encap:Ethernet HWaddr 08:00:27:89:B3:27
inet addr:10.0.2.15 Bcast:0.0.0.0 Mask:255.255.255.0 inet6 addr:
fe80::a00:27ff:fe89:b327/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:303 errors:0 dropped:0 overruns:0 frame:0 TX packets:276
errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000
RX bytes:34984 (34.1 KiB) TX bytes:37717 (36.8 KiB)
lo
 Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:0 errors:0
dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

Desactivación controlador USB

La máquina virtual proporcionada tiene el soporte USB 2.0 desactivado. Si por algún motivo no arranca dando un mensaje de error referente al controlador USB verificar que está desactivado.



En otros casos es necesario descargar e instalar el Oracle VM VirtualBox Extension Pack que se puede descargar en la misma página de downloads de VirtualBox.

Activación del soporte para la virtualización

Si la máquina virtual no arranca una posible fuente de problemas es no tener el soporte hardware de virtualización activado. La máquina proporcionada contiene un sistema operativo de 64 bits, por lo que es necesario el soporte hardware para la virtualización (Intel VT-x o AMD-v). En esta situación hay que revisar los siguientes puntos (si la máquina ya ha arrancado lo podéis ignorar):

1. La CPU del ordenador debe tener la capacidad y el soporte para la tecnología de virtualización Intel o AMD: VT-x o AMD-v. Ante la duda revisar el modelo exacto de CPU y validar que tiene capacidad de 64 bits y soporte de virtualización. La mayoría de los ordenadores actuales tienen este soporte.

2. Hay que tener activado VT-x / AMD-v en la BIOS del ordenador. Hay que entrar en la configuración de la BIOS y buscar en uno de los menús una entrada del tipo "Enable Virtualization Technology". Después de hacer el cambio de configuración de la BIOS hay que reiniciar el equipo.

3. También debemos asegurarnos de que no hay ninguna aplicación de la máquina anfitrión que utiliza VT-x / AMD-v. Podemos encontrar "system level debuggers", otras plataformas de virtualización o algunas aplicaciones anti-virus residentes. Estos problemas suelen ser habituales cuando el sistema operativo host (o anfitrión) es un Windows 64bit ya que estos a menudo activan la plataforma Hyper-v de Microsoft por defecto. Hyper-v toma el control de VT-x y no dejará funcionar correctamente en VirtualBox.

Uso de MPI en Python

Una vez ya hemos instalado y arrancado la máquina virtual, es posible crear y ejecutar aplicaciones MPI.

Mediante línea de comandos

Para ello nos conectamos a la máquina virtual (mediante *ssh* o mediante la consola). A continuación, se ejemplifica la ejecución de una aplicación MPI. En el siguiente ejemplo, el proceso 0 envía dos mensajes al proceso 1:

`mpiExample.py`

```
from mpi4py import MPI
import numpy

comm = MPI.COMM_WORLD
rank = comm.Get_rank()

# passing MPI datatypes explicitly
if rank == 0:
    data = numpy.arange(1000, dtype='i')
    comm.Send([data, MPI.INT], dest=1, tag=77)
elif rank == 1:
    data = numpy.empty(1000, dtype='i')
    comm.Recv([data, MPI.INT], source=0, tag=77)

# automatic MPI datatype discovery
if rank == 0:
    data = numpy.arange(100, dtype=numpy.float64)
    comm.Send(data, dest=1, tag=13)
elif rank == 1:
    data = numpy.empty(100, dtype=numpy.float64)
    comm.Recv(data, source=0, tag=13)
```

MPI para Python

Librería matemática

Para lanzar una aplicación paralela con MPI, normalmente se utiliza el comando “mpirun” o “mpiexec”. Para conocer un poco más estos parámetros podéis entrar con el parámetro de ayuda (--help) de “mpirun”.

```
sd-GCDA:~# mpirun --help
mpirun (Open MPI) 4.0.5

Usage: mpirun [OPTION]...  [PROGRAM]...
Start the given program using Open RTE

-cl-npl--np <arg0>       Number of processes to run
   -gmcal--gmca <arg0> <arg1>
                        Pass global MCA parameters that are applicable to
                        all contexts (arg0 is the parameter name; arg1 is
                        the parameter value)
-hl--help <arg0>         This help message
   -mcal--mca <arg0> <arg1>
                        Pass context-specific MCA parameters; they are
                        considered global if --gmca is not used and only
                        one context is specified (arg0 is the parameter
                        name; arg1 is the parameter value)
   -nl--n <arg0>         Number of processes to run
-ql--quiet              Suppress helpful messages
-vl--verbose             Be verbose
-Vl--version             Print version and exit
```

Finalmente, para ejecutar el ejemplo anterior, se hace de la siguiente manera:

```
sd-GCDA:~# mpirun -np 2 --host localhost:2 python3.9 ./mpiExample.py
```

Donde:

-np: es el número de procesos

-host: es la máquina o host en donde se pide ejecutar e indica el número de slots disponibles (CPU/core) que tiene el host para ejecutar.

-python3.9 ./mpiExample.py: esta sería la aplicación a ejecutar.