



Unidad 3 / Escenario 6

Instructivo de procesamiento HPC

# Instructivo de procesamiento sobre HPC

## Contenido

### 1 Proceso de programación paralela

---

## Notas importantes

**Orden de encendido de un sistema HPC:** primero se enciende el máster, luego el nodo cero, después el nodo 1 y así sucesivamente.

**Orden de apagado:** se apagan primero los nodos y luego el máster.

## Proceso de programación paralela

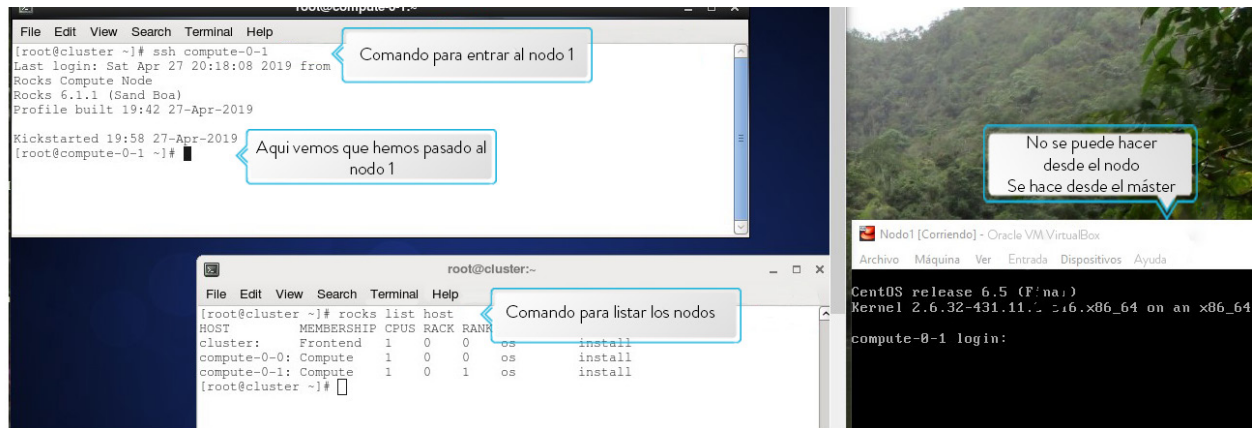
1. Para trabajar en Rocks, hay que revisar el manual de comandos y aprender a usarlos. Por ejemplo, para entrar al Nodo1, desde el máster, se lista primero el conjunto de nodos esclavos con el comando `rocks list host`, como se muestra en la figura 1, luego se ejecuta el comando `ssh`.

```
File Edit View Search Terminal Help
[root@cluster ~]# rocks list host
HOST      MEMBERSHIP CPUS RACK RANK RUNACTION INSTALLACTION
cluster:  Frontend  1    0    0    os      install
compute-0-0: Compute  1    0    0    os      install
compute-0-1: Compute  1    0    1    os      install
[root@cluster ~]#
```

**Figura 1. Consola de listado de nodos**

Fuente: elaboración propia

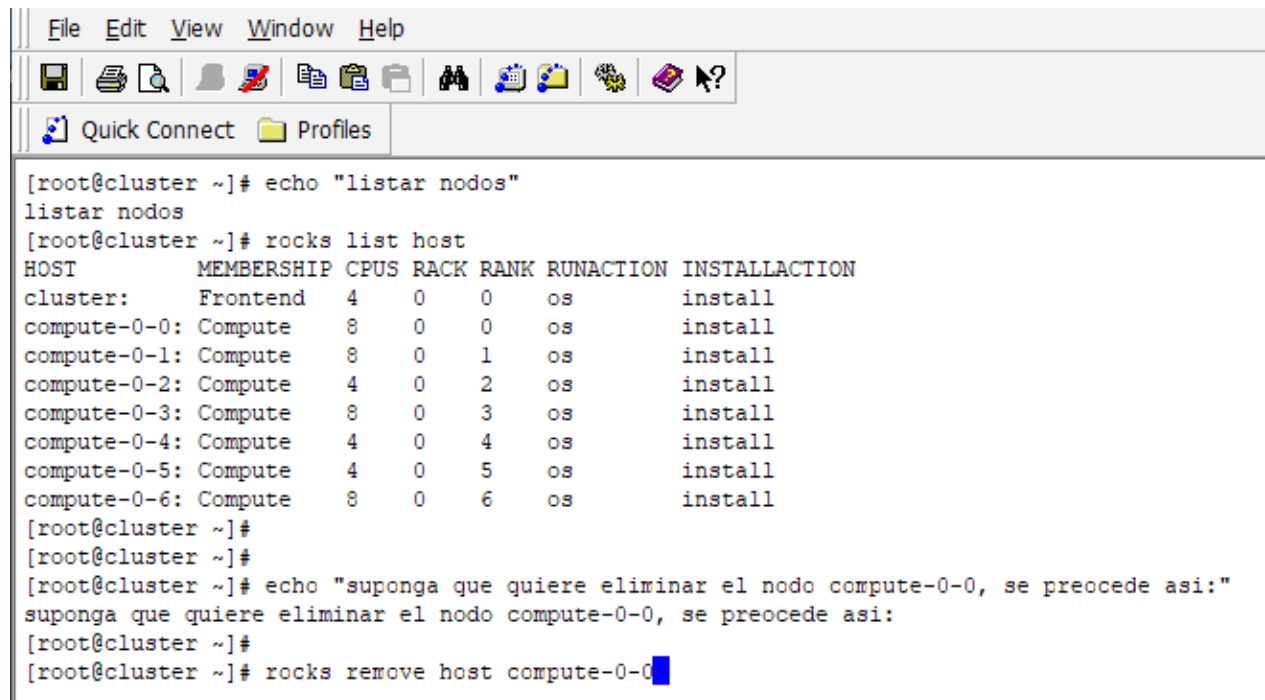
Conexión al nodo.



**Figura 2. Consola de conexión al Nodo1**

Fuente: elaboración propia

2. Para borrar un nodo se elimina de VirtualBox y, posteriormente, se remueve del sistema



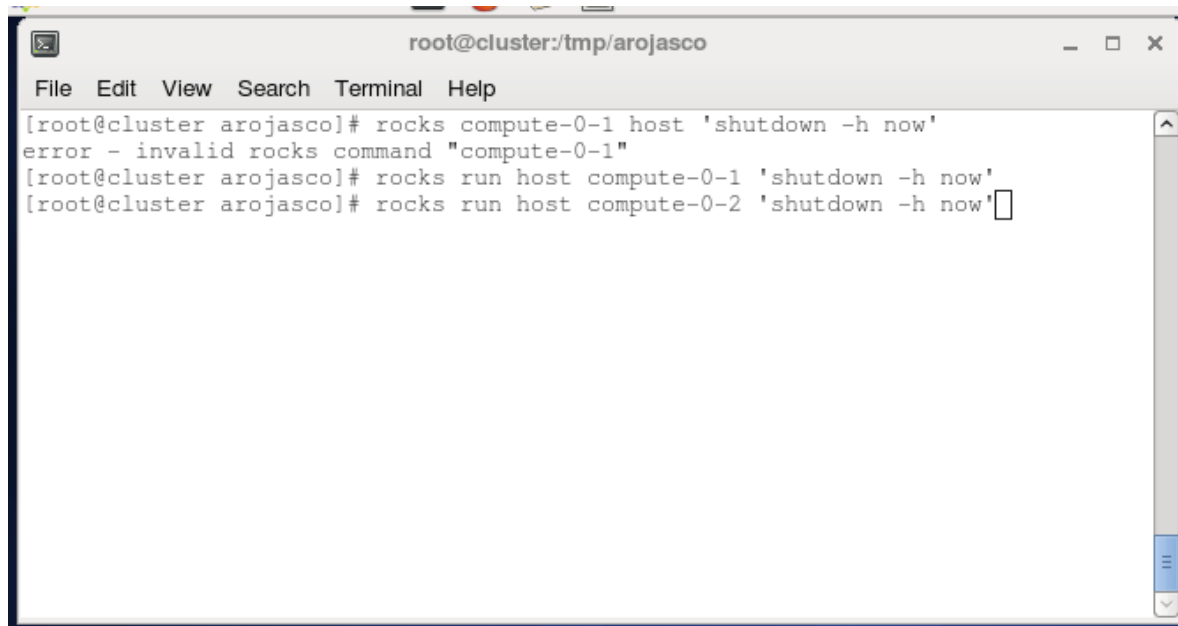
**Figura 3. Consola para mostrar cómo se remueve un nodo**

Fuente: elaboración propia

Para volver a crear un nodo se carga primero el *insert-ethers*.

3. Para apagar nodos desde el máster sin necesidad de entrar al nodo, se puede apagar utilizando uno de los comandos de Rocks para apagado masivo. Se puede elaborar una Shell en el máster y cada vez que se requieran apagar todos los nodos se ejecuta la Shell.

El comando es: ***rocks run host compute-0-0'shutdown -h now'***.



```
root@cluster:/tmp/arojasco
File Edit View Search Terminal Help
[root@cluster arojasco]# rocks compute-0-1 host 'shutdown -h now'
error - invalid rocks command "compute-0-1"
[root@cluster arojasco]# rocks run host compute-0-1 'shutdown -h now'
[root@cluster arojasco]# rocks run host compute-0-2 'shutdown -h now'
```

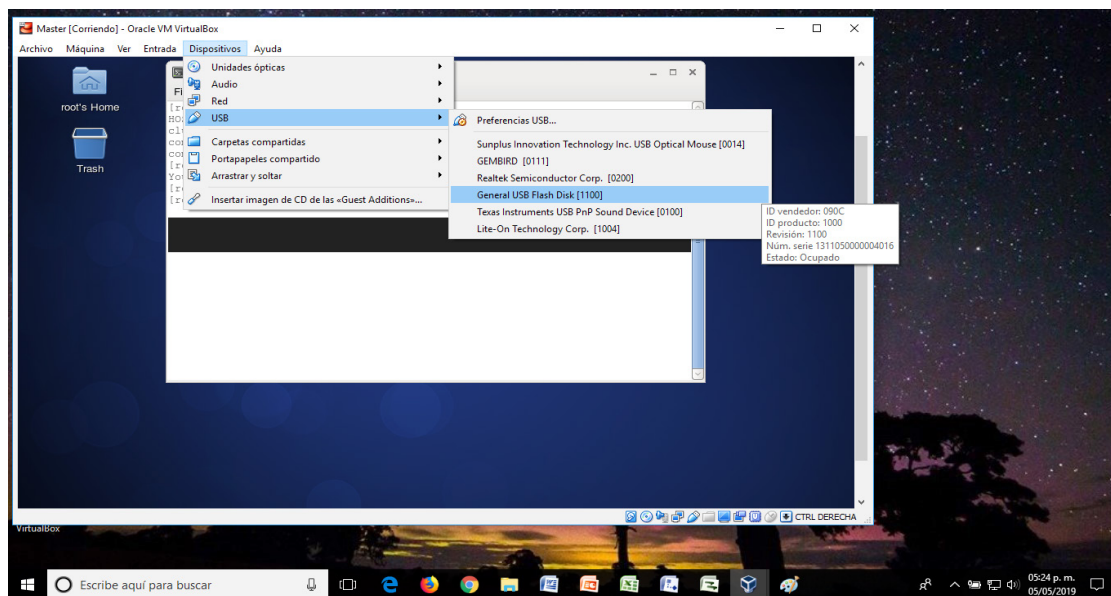
**Figura 4. Consola de apagado masivo de nodos desde el máster**

Fuente: Elaboración propia

**Nota:** Para ingresar a los nodos, no se puede desde el login del nodo. Hay que hacerlo con ssh, desde el máster, a través de una consola.

En caso de que no haya sido posible instalar internet por problemas con el *firewall* del lugar en el que se encuentra o porque el antivirus bloquee la salida, se puede habilitar la lectura desde USB, para lo cual se debe configurar. La USB debe estar formateada en FAT32.

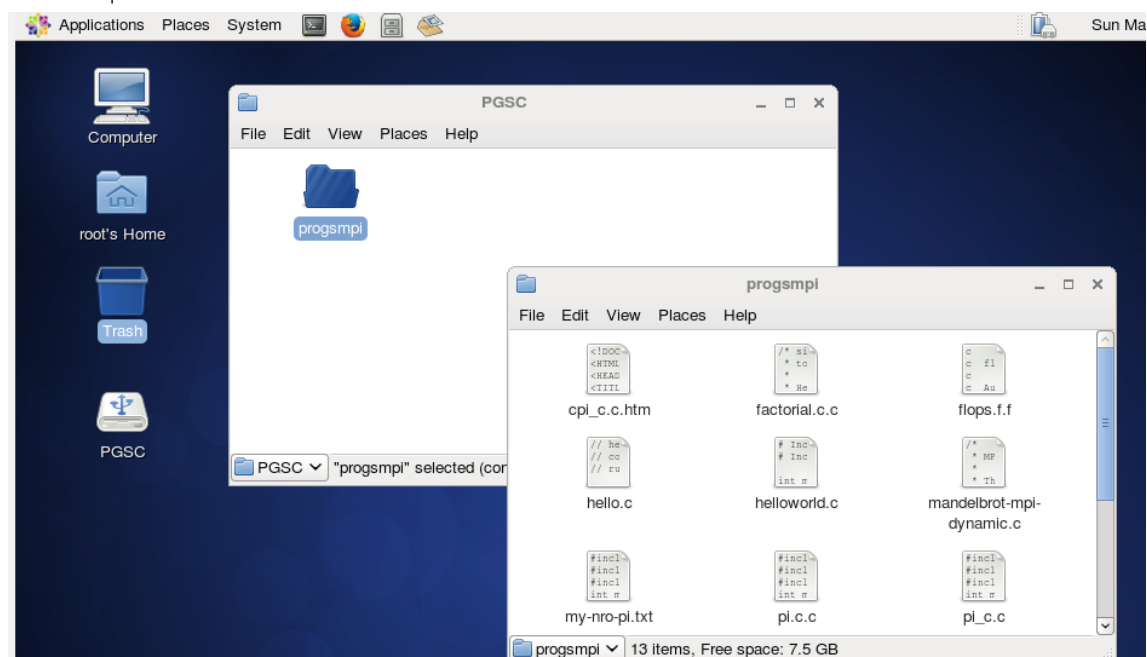




**Figura 5. Consola de configuración de la USB en el máster**

Fuente: Elaboración propia

4. Para cargar la USB, debe colocar el máster en modo pantalla completa y luego sí ingresar por dispositivos, USB, General USB.



**Figura 6. Consola de configuración de la USB en el máster**

Fuente: elaboración propia

5. Una vez la USB es reconocida, se procede a copiar los programas a un directorio, como por ejemplo en /tmp/programas. En este punto, se puede comenzar a procesar en paralelo haciendo uso de un sistema operacional distribuido con aplicaciones paralelas distribuidas. A continuación, se muestra un ejemplo para cargar un programa en Fortran, tenga en cuenta que aquellos que inician con c son comentarios y se resalta en azul; mientras que, en amarillo se resalta lo que va a correr en los nodos:

```
c
c flops.f
c
c Author:
c Yuri Sbitnev <yuri@linux-ekb.info>
c
c Copyright (c) 2008-2009 Yuri Sbitnev
c
c This program is free software; you can redistribute it and/or
c modify
c it under the terms of the GNU General Public License as published
c by
c the Free Software Foundation; either version 2 of the License, or
c (at your option) any later version.
c
c This program is distributed in the hope that it will be useful,
c but WITHOUT ANY WARRANTY; without even the implied warranty of
c MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
c GNU General Public License for more details.
c
c You should have received a copy of the GNU General Public License
c along with this program; if not, write to the Free Software
c Foundation, Inc., 59 Temple Place, Suite 330, Boston,
c MA 02111-1307 USA
```



```

program calc_mflops
  include 'mpif.h'
  integer i, j, n
  double precision w, gsum, sum
  double precision v
  integer np, myid, ierr, niter, status(MPI_STATUS_SIZE)
  real*8 time, amflops, amflops1, time1, time2, dsecnd
  integer mflops, mflops1
c Initialize MPI. Find number of processors.
  call MPI_INIT( ierr )
  call MPI_COMM_RANK( MPI_COMM_WORLD, myid, ierr )
  call MPI_COMM_SIZE( MPI_COMM_WORLD, np, ierr )
c Zero process determines the number of points.
  if ( myid .eq. 0 ) then
    n = 200000000
  endif

  time1 = MPI_Wtime()

c Send number of point from zero process to all other processes
  call MPI_BCAST(n, 1, MPI_INTEGER, 0, MPI_COMM_WORLD, ierr)
c Calculate partial sum
  w = 1.0 / n

  do j = 1, 4
    sum = 0.0d0
    do i = myid+1, n, np
      v = (i - 0.5d0) * w

```



```
v = 4.0d0 / (1.0d0 + v * v)
```

```
sum = sum + v
```

```
end do
```

```
end do
```

c Summarize the partial sums and store it in zero process

```
call MPI_REDUCE(sum, gsum, 1, MPI_DOUBLE_PRECISION,
```

```
$ MPI_SUM, 0, MPI_COMM_WORLD, ierr)
```

```
time2 = MPI_Wtime()
```

```
time = (time2 - time1) / 4
```

```
niter = 0
```

```
do i = myid+1, n, np
```

```
niter = niter + 1
```

```
end do
```

```
mflops1 = 9 * niter / (1000000.0 * time)
```

c Zero process prints cluster benchmark results

```
if (myid .eq. 0) then
```

```
mflops = 9 * n / (1000000.0 * time)
```

```
print *, ' '
```

```
print '(A)', ' HPC Test -----'
```

```
print '(A,I2.1)', ' Quantity of processors = ', np
```

```
print '(A,F6.2,A)',
```

```
$' Calculation time = ', time, ' seconds'
```

```
print '(A,I6.1,A)',
```

```
$' Cluster speed = ', mflops, ' MFLOPS'
```





```

print '(A)', ' -----',
print '(A,I2.2,A,I6.1,A)',
$' Cluster node N',0,' speed = ', mflops1, ' MFLOPS'
c Collect and print benchmark results from individual processes
do i = 1, np-1
CALL MPI_RECV(mflops1, 1, MPI_REAL8, i, 0,
$ MPI_COMM_WORLD, status, ierr)
print '(A,I2.2,A,I6.1,A)',
$' Cluster node N', i, ' speed = ', mflops1, ' MFLOPS'
end do
print '(A)', ' -----',
print *, ' \
else
c Send local process benchmark result to zero process
call MPI_SEND(mflops1, 1, MPI_REAL8, 0, 0,
$ MPI_COMM_WORLD, ierr)
endif

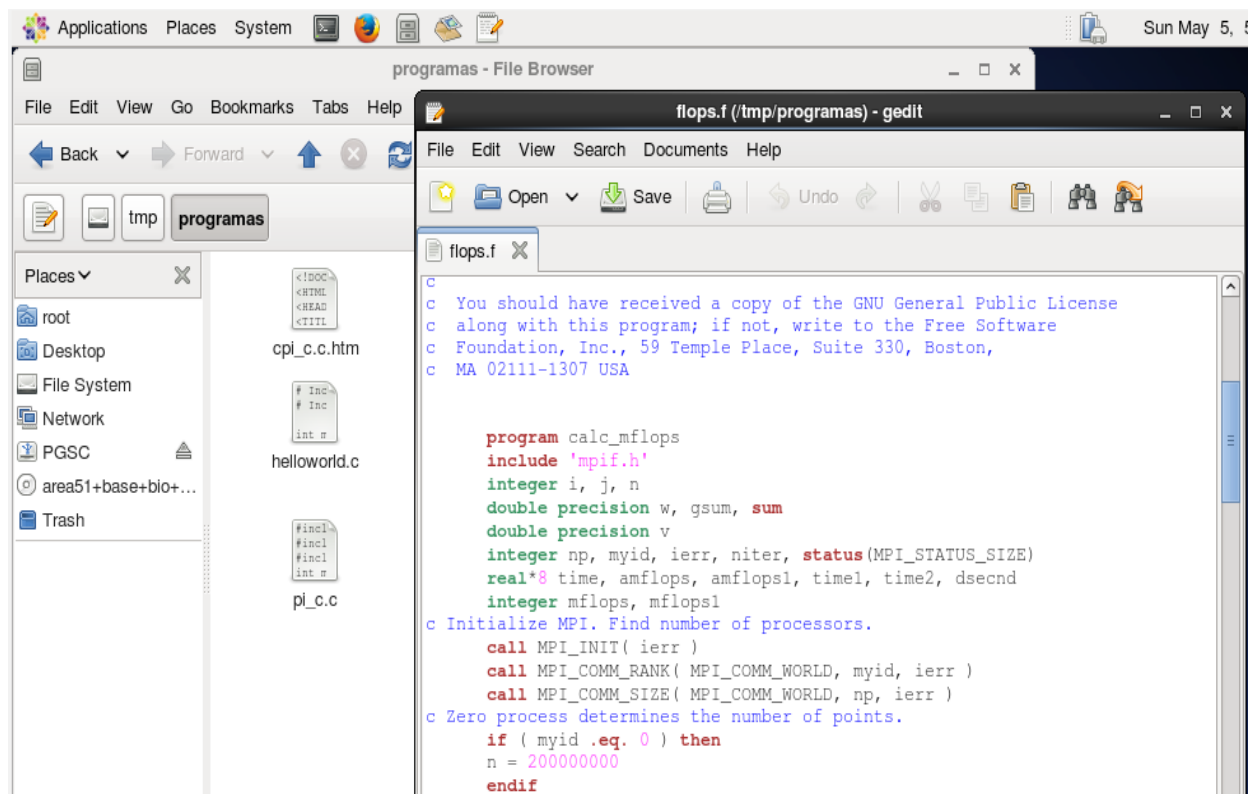
c Close MPI
call MPI_FINALIZE(ierr)
end

```

Este programa se dirige a cada nodo y revisa cuántos ciclos de reloj hace por segundo. Este proceso se llama *flops*, por lo tanto el programa se llama *flops.f*

Se puede editar (con el editor de Linux llamado *gedit*) en el máster entrando por aplicaciones.





**Figura 7. Un segmento del programa flops.f editado en el máster**

Fuente: elaboración propia

Tenga en cuenta:

Para compilar en Fortran-77 el comando es: ***mpif77 flops.f -o ciclos.***

- En la figura 8 se muestra el proceso de compilación del programa flops.f en el que el objeto se llamó ciclos y se ejecuta con el comando ***mpirun -np 2 ciclos.***, donde np es el número de servidores por el que va a correr, en este caso con los nodos 0 y 1.

```
Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Applications Places System
root@cluster:/tmp/programas
File Edit View Search Terminal Help
[root@cluster programas]# ls -l *.f
-rwxr-xr-x 1 root root 3349 Aug 30 2012 flops.f
-rwxr-xr-x 1 root root 711 Oct 7 2013 prueba.f
[root@cluster programas]#
[root@cluster programas]# mpif77 flops.f -o cliclos
[root@cluster programas]#
[root@cluster programas]# ls -ltr
total 80
-rwxr-xr-x 1 root root 2575 Aug 30 2001 cpi_c.c.htm
-rwxr-xr-x 1 root root 10192 Apr 7 2011 mandelbrot-mpi-dynamic.c
-rwxr-xr-x 1 root root 1132 Aug 28 2012 pi.c.c
-rwxr-xr-x 1 root root 323 Aug 28 2012 helloworld.c
-rwxr-xr-x 1 root root 3349 Aug 30 2012 flops.f
-rwxr-xr-x 1 root root 1922 Aug 30 2012 factorial.c.c
-rwxr-xr-x 1 root root 4819 Aug 30 2012 prime_mpi.cpp
-rwxr-xr-x 1 root root 630 Aug 30 2012 hello.c
-rwxr-xr-x 1 root root 711 Oct 7 2013 prueba.f
-rwxr-xr-x 1 root root 1167 Apr 1 2014 my-nro-pi.txt
-rwxr-xr-x 1 root root 1163 May 3 11:03 pi.c.c
-rwxr-xr-x 1 root root 4822 May 3 11:11 prime_mpi.c
-rwxr-xr-x 1 root root 14444 May 5 18:13 cliclos
[root@cluster programas]#
```

**Figura 8. Consola de compilación de un programa en el máster**

Fuente: elaboración propia

```
Master [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
Applications Places System Sun May 5, 6:34

root@cluster:/tmp/programas
File Edit View Search Terminal Help
[root@cluster programas]# ls -l *.f
-rwxr-xr-x 1 root root 3349 Aug 30 2012 flops.f
-rwxr-xr-x 1 root root 711 Oct 7 2013 prueba.f
[root@cluster programas]# mpif77 flops.f -o ciclos
[root@cluster programas]# rocks list host
HOST      MEMBERSHIP CPUS RACK RANK RUNACTION INSTALLACTION
cluster:  Frontend 1    0    0    os      install
compute-0-0: Compute 1    0    0    os      install
compute-0-1: Compute 1    0    1    os      install
[root@cluster programas]#
[root@cluster programas]# mpirun -np 2 ciclos

HPC Test -----
Quantity of processors = 2
Calculation time      = 3.55 seconds
Cluster speed         = 506 MFLOPS

Cluster node N00 speed = 253 MFLOPS
Cluster node N01 speed = 253 MFLOPS

[root@cluster programas]#
```

Compilation: `mpif77 flops.f -o ciclos`

Execution: `mpirun -np 2 ciclos`

Results:

Host	Membership	CPUs	Rack	Rank	RunAction	InstallAction
cluster:	Frontend	1	0	0	os	install
compute-0-0:	Compute	1	0	0	os	install
compute-0-1:	Compute	1	0	1	os	install

HPC Test Results:

Metric	Value
Quantity of processors	2
Calculation time	3.55 seconds
Cluster speed	506 MFLOPS
Cluster node N00 speed	253 MFLOPS
Cluster node N01 speed	253 MFLOPS

**Figura 9. Consola de ejecución y resultados de una corrida del programa ciclos desde el máster ejecutado en los nodos**  
Fuente: elaboración propia

Veamos ahora un segundo ejemplo con C++.

```
// hello.c - Hello World with MPI
// compile: mpicc -Wall -O -o hello hello.c
// run: mpirun -np num_procs hello
```



```

#include <stdio.h>
#include <mpi.h>

int main(int argc, char ** argv)
{
    int p; // size
    int id; // rank

    MPI_Init(&argc, &argv); // start up "virtual machine"
    MPI_Comm_size(MPI_COMM_WORLD, &p); // get size of VM
    MPI_Comm_rank(MPI_COMM_WORLD, &id); // get own rank in VM

    printf("Hola Luz Adriana. Yo soy Alexis Rojas y te saludo desde
el nodo %d of %d\n", id, p); // payload

    MPI_Finalize(); // shut down VM
    return 0;
}

```

Este programa va a cada nodo, toma por ejemplo su cédula o identificación y la envía al máster con el siguiente aviso: hola señor máster, yo soy, el nodo número x de un total de N nodos.

Se **compila** con el siguiente comando: ***mpicc hola.c -o holamundo***

Se **ejecuta** con mpirun, así: ***mpirun -np 2 holamundo***



```

root@cluster:/tmp/programas
File Edit View Search Terminal Help
[root@cluster programas]# ls -l *.c
-rwxr-xr-x 1 root root 1922 Aug 30 2012 factorial.c.c
-rwxr-xr-x 1 root root 630 Aug 30 2012 hello.c
-rwxr-xr-x 1 root root 323 Aug 28 2012 helloworld.c
-rwxr-xr-x 1 root root 10192 Apr 7 2011 mandelbrot-mpi-dynamic.c
-rwxr-xr-x 1 root root 1163 May 3 11:03 pi.c.c
-rwxr-xr-x 1 root root 1132 Aug 28 2012 pi.c.c
-rwxr-xr-x 1 root root 4822 May 3 11:11 prime_mpi.c
[root@cluster programas]# mv hello.c hola.c
[root@cluster programas]# gedit hola.c
[root@cluster programas]#
[root@cluster programas]# mpicc hola.c -o holamundo
[root@cluster programas]# ls -l hola*
-rwxr-xr-x 1 root root 640 May 5 18:44 hola.c
-rw-r--r-- 1 root root 630 Aug 30 2012 hola.c~
-rwxr-xr-x 1 root root 9208 May 5 18:44 holamundo
[root@cluster programas]#
[root@cluster programas]# mpirun -np holamundo
-----
No executable was specified on the mpirun command line.

Aborting.

-----
[root@cluster programas]# mpirun -np 2 holamundo
Hola Luz Adriana. Yo soy el nodo 1 de un total de 2
Hola Luz Adriana. Yo soy el nodo 0 de un total de 2
[root@cluster programas]#

```

**Figura 10. Consola de ejecución y resultados de una corrida del programa holamundo desde el máster ejecutado en los nodos**

Fuente: elaboración propia

Para la edición de programas se utiliza vi, pero si no está familiarizado con este programa puede utilizar gedit.

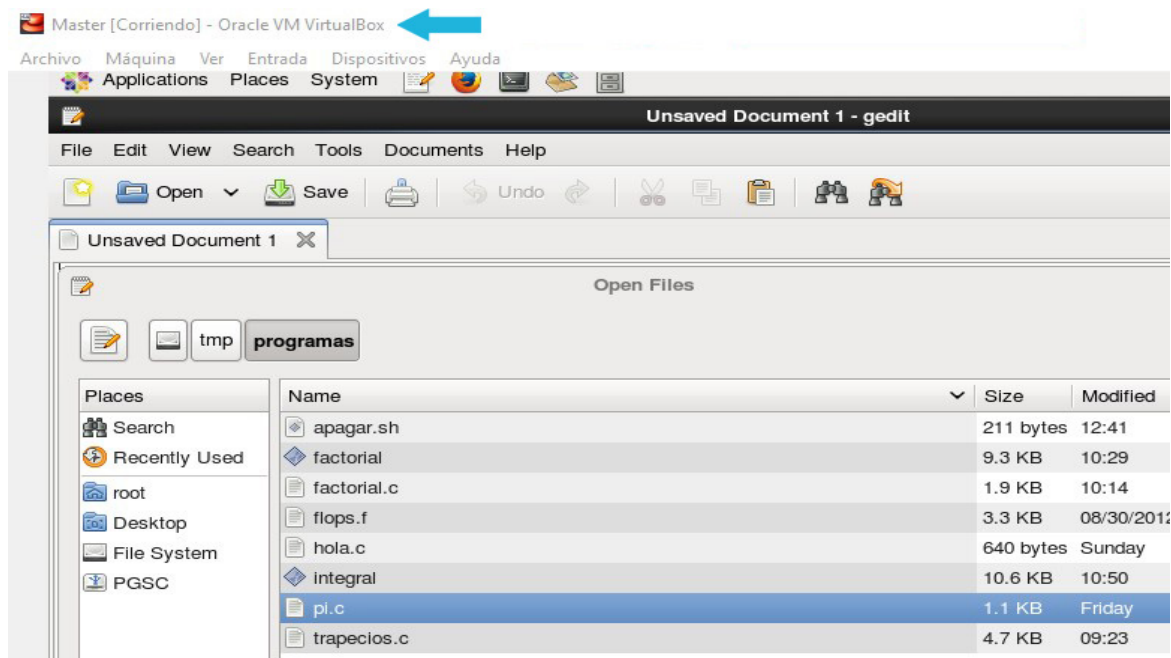
```

root@cluster:/tmp/programas
Search Terminal Help
[gramas]# ls -ltr
root 3349 Aug 30 2012 flops.f
root 1163 May 3 11:03 pi.c
root 640 May 5 18:44 hola.c
root 4830 May 9 09:23 trapecios.c
-rw-r--r-- 1 root root 1927 May 9 10:14 factorial.c
-rwxr-xr-x 1 root root 9474 May 9 10:29 factorial
-rwxr-xr-x 1 root root 10814 May 9 10:50 integral
-rwxrwxrwx 1 root root 211 May 9 12:41 apagar.sh
[root@cluster programas]#

```

**Figura 11. Consola de ejecución del editor de texto gedit**

Fuente: elaboración propia



**Figura 12. Consola de invocación edición del programa pi.c**

Fuente: elaboración propia

Composición del clúster, compilación, ejecución y resultados.

```
File Edit View Search Terminal Help
[root@cluster programas]# ls -l
total 40
-rwxrwxrwx 1 root root 211 May 9 12:41 apagar.sh
-rwxr-xr-x 1 root root 9474 May 9 10:29 factorial
-rw-r--r-- 1 root root 1927 May 9 10:14 factorial.c
-rwxr-xr-x 1 root root 3349 Aug 30 2012 flops.f
-rwxr-xr-x 1 root root 640 May 5 18:44 hola.c
-rwxr-xr-x 1 root root 1141 May 9 13:08 pi.c
-rwxr-xr-x 1 root root 4830 May 9 09:23 trapecios.c
[root@cluster programas]#
[root@cluster programas]# mpicc pi.c -o PI
[root@cluster programas]#
[root@cluster programas]#
[root@cluster programas]# mpirun -np 2 ./PI
10
entre el numero de intervalos: (0 quits) pi es aproximadamente 3.142425985001098
3, con un error de 0.0008333314113051
20
entre el numero de intervalos: (0 quits) pi es aproximadamente 3.1418009868930934,
on un error de 0.000208333033003
0
entre el numero de intervalos: (0 quits) [root@cluster programas]#
```

**Figura 13. Consola de del programa pi.c**

Fuente: elaboración propia



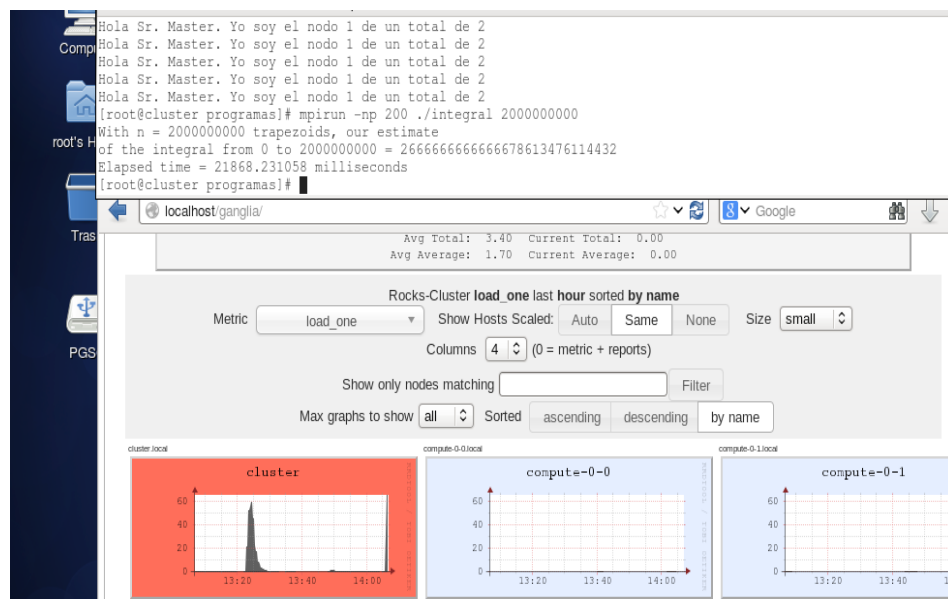


## 7. Ejecución del programa del cálculo del área, mediante el proceso de integración

```
root@cluster:/tmp/programas
File Edit View Search Terminal Help
[root@cluster programas]# ls -l
total 52
-rwxrwxrwx 1 root root 211 May  9 12:41 apagar.sh
-rwxr-xr-x 1 root root 9474 May  9 10:29 factorial
-rw-r--r-- 1 root root 1927 May  9 10:14 factorial.c
-rwxr-xr-x 1 root root 3349 Aug 30 2012 flops.f
-rwxr-xr-x 1 root root 640 May  5 18:44 hola.c
-rwxr-xr-x 1 root root 9466 May  9 13:12 PI
-rwxr-xr-x 1 root root 1141 May  9 13:08 pi.c
-rwxr-xr-x 1 root root 4830 May  9 09:23 trapecios.c
[root@cluster programas]#
[root@cluster programas]#
[root@cluster programas]#
[root@cluster programas]# mpicc trapecios.c -o integral
[root@cluster programas]#
[root@cluster programas]#
[root@cluster programas]# mpirun -np 2 ./integral 200
With n = 200 trapezoids, our estimate
of the integral from 0 to 2000000000 = 26666999999998095593046016
Elapsed time = 0.006199 milliseconds
[root@cluster programas]# mpirun -np 200 ./integral 2000000000
```

**Figura 14. Consola de visualización de la ejecución de una integral**

Fuente: elaboración propia



**Figura 15. Consola de visualización de carga**

Fuente: elaboración propia



Los nodos empiezan a trabajar con la ejecución del programa de la integral entre (a,b) donde  $a = 0$  y  $b = 2000.000.000$ . y 2000 millones de trapecios.

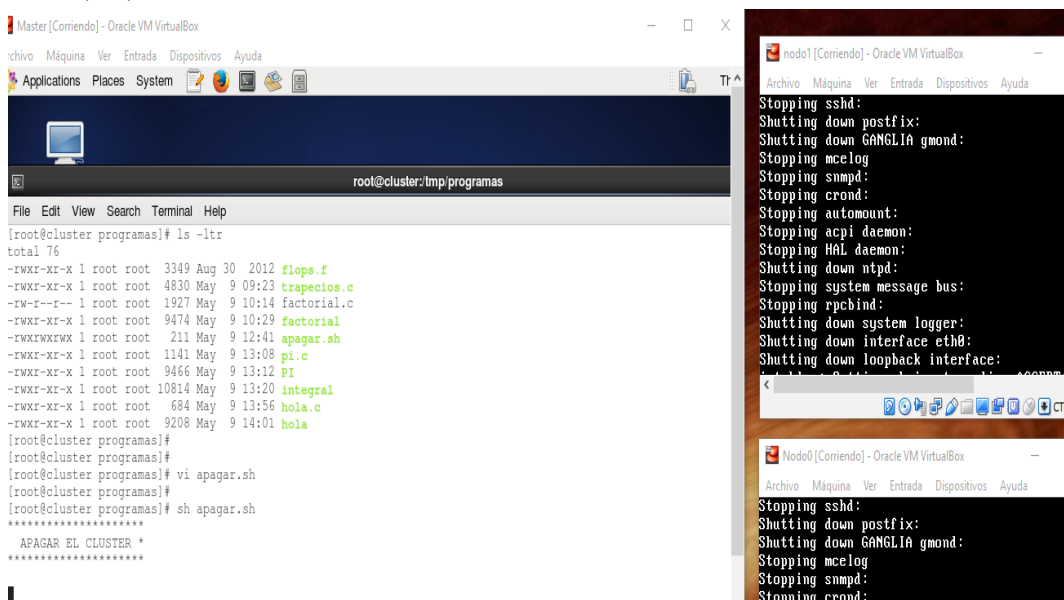
Finalmente, se procede a apagar el sistema.

```
root@cluster:/tmp/programas
File Edit View Search Terminal Help
echo '*****'
echo '  APAGAR EL CLUSTER *'
echo '*****'
echo ' '
rocks run host compute-0-0 'shutdown -h now'
rocks run host compute-0-1 'shutdown -h now'
sleep 15
shutdown -h now
```

**Figura 16. Shell de ejecución del apagado masivo desde el máster**

Fuente: elaboración propia

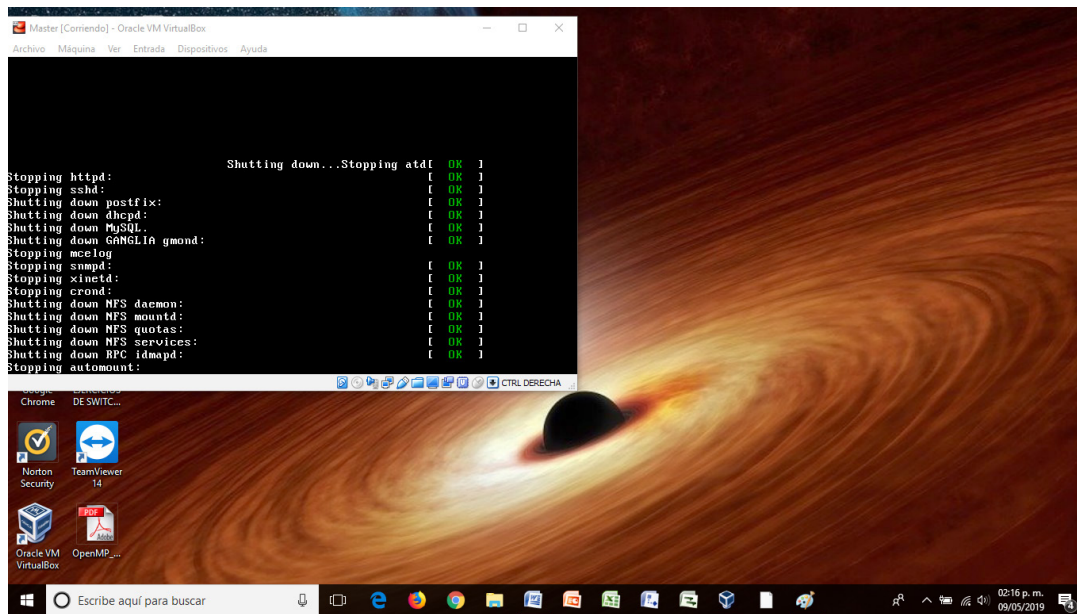
Los nodos y el máster se deben apagar correctamente para que no se dañe el sistema y se tenga que repetir el proceso de instalación y puesta a punto. En este caso, se usó una Shell desarrollada para dicho propósito.



**Figura 17. Shell de ejecución del apagado masivo desde el máster**

Fuente: elaboración propia





**Figura 18. Cierre final del máster**

*Fuente: elaboración propia*

# Referencias

Atlassian. (s.f.). Confluence. Recuperado de <https://www.nersc.gov/users/software/programming-models/>

MPI. (s.f.). C++ *Examples*. Recuperado de [https://people.sc.fsu.edu/~jburkardt/cpp\\_src/mpl/mpl.html](https://people.sc.fsu.edu/~jburkardt/cpp_src/mpl/mpl.html)

Nersc. (2014). *IOR*. Recuperado de <https://www.nersc.gov/users/computational-systems/cori/nersc-8-procurement/trinity-nersc-8-rfp/nersc-8-trinity-benchmarks/ior/>



## INFORMACIÓN TÉCNICA



FACULTAD DE  
**INGENIERÍA, DISEÑO  
E INNOVACIÓN**

**Módulo:** Sistemas Distribuidos

**Unidad 3:** Computación en clúster y programación paralela.

**Escenario 6:** Procesamiento y computación paralela

**Autor:** Alexis Rojas Cordero

**Asesor Pedagógico:** Jeimy Lorena Romero Perilla

**Diseñador Gráfico:** Nicolás Jiménez Osorio

**Asistente:** María Elizabeth Avilán Forero

*Este material pertenece al Politécnico Gran Colombiano.*

*Prohibida su reproducción total o parcial.*