"Super PI": Clúster mejorado y optimizado para menor consumo de energía con Raspberry Pi3

César Martín Cruz Salazar† & Yuri Nuñez Medrano
Escuela de Ciencia de la Computación - Facultad de Ciencias.
Universidad Nacional de Ingeniería;
†*ccruz@uni.edu.pe*

Recibido el 30 de Mayo del 2016; aceptado el 14 de Junio del 2016

Este trabajo de investigación usa minicomputadores Raspberry Pi3 basados en ARM con el fin de construir un clúster llamado "Super PI" potente y optimizado para bajo consumo energético y bajo costo. Se ha dado especial énfasis en obtener un clúster de una potencia de procesamiento que sea mayor a las computadoras PCs que se usan en nuestra universidad. Quisimos demostrar que aumentando el número de nodos logramos aumentar en forma considerable la potencia de computo del clúster obteniendo de esta manera una máquina potente en comparación con otras máquinas. El resultado de las comparaciones del clúster con computadoras PCs comunes son mostrados aquí. Al mismo tiempo que se construía este clúster, se construía otro clúster con placas Odroid XU4 basados en ARM. Se muestran, también las comparaciones con este clúster en este artículo.

Palabras Claves: clúster, raspberry pi versiones 2 y 3, programación paralela, programación distribuida, clúster educativo, odroid xu4.

This research work uses Raspberry Pi3 minicomputers based in ARM in order to build a powerful and optimized for low energy consumption and low cost cluster called "Super PI". Special emphasis has been placed on obtaining a cluster of processing power that is larger than PC computers that are used in our university. We wanted to show that increasing the number of nodes the computing power of the cluster considerably increased obtaining thus a powerful machine compared to PCs. The result of these cluster comparisons with PC computers are shown here. At the same time that this cluster was being built, another cluster with Odroid XU4 boards based in ARM was built. We show the comparisons of computing power with this one in this paper.

Keywords: cluster, raspberry pi version 2 and 3, parallel programming, distributive programming, educational cluster, odroid xu4.

1 Introducción

Nuestro grupo tiene ya una acumulada experiencia en la construcción de clústeres [1, 2], aunque estos no lograrón obtener una potencia de computo ni siquiera mayor a las computadoras PCs; es por esto que quisimos esta vez ir mas allá en la potencia de procesamiento. Esta vez, pusimos hincapié en construir un clúster(figura 2) que supere en potencia de procesamiento a las computadoras PC comunes que se utilizan en nuestra Universidad. Este clúster, también debía superar al "Clúster CTIC" construido con máquinas PCs recicladas [3]. Es así, que en ese intento construimos un clúster con 29 nodos equivalente a 116 núcleos. De las cuales un nodo(4 núcleos) es para el maestro que corresponde a una placa Raspberry Pi2 y los demás nodos(28) son los esclavos que corresponden a placas Raspberry Pi3 (figura 1). En el mismo tiempo, que se construía el Súper PI se construía otro clúster con un objetivo distinto usando en este caso 20 placas Odroid XU4 [4]. La construcción de este clúster será descrito más en detalle en otro artículo.

2 Especificaciones del clúster Súper PI

Proporcionamos una descripción de sus componentes de hardware y del software instalado.

2.1 Hardware

El clúster se construyó con una configuración maestro/esclavo (figura 3). Para el nodo maestro se usó un Raspberry Pi versión 2 y para los nodos esclavos se usó un Raspberry Pi(RPI) versión 3. Estas placas son del tamaño de una tarjeta de crédito, cada una cuenta con 1 Gigabyte de RAM, un sistema sobre un Chip(SoC, System on a Chip) Broadcom BCM2837(RPI3) y Broadcom BCM2836(RPI2), que integra un procesador quad core ARM Cortex-A53 de 1.2GHz, una arquitectura de 64 bits, un GPU Broadcom VideoCore IV con salida HDMI y salida de vídeo RCA, y un controlador USB. La pl también contiene, un adaptador para puerto Ethernet externo de 10/100 Mbits, y un adaptador de cuatro puertos USB 2.0. En la tabla 1 se muestra un resumen d las especificaciones del RPI3. El almacenamiento local del sistema operativo y archivos de datos se colocó e una tarjeta de memoria microSD de 32 gigabytes cl 10(figura 5) que se coloca en una ranura acondicionad

en la placa. La placa RPI3 se alimenta con un adaptador de voltaje de 5V a 3A. Se consideró necesario instalar varios ventiladores iguales a los mostrados en la (figura 4). Sin estos ventiladores la temperatura que se tenía era 45 grados centigrados en estado standby y 84 grados durante la ejecución de un programa en todos los nodos. Con los ventiladores la temperatura bajó a 39 grados centigrados en estado standby y 76 grados en promedio durante la ejecución de un programa.

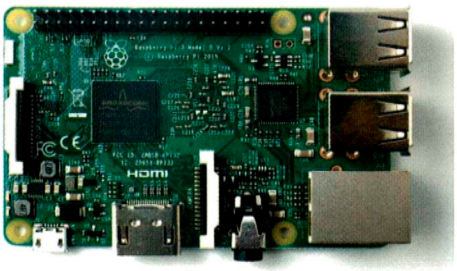

Figure 1. *Raspberry Pi 3 modelo B*

Tabla 1. Especificaciones para Raspberry pi 3 modelo B.

SoC	Broadcom BCM2837
SoC	Broadcom BCM2837
CPU	1.2GHz quad-core ARM Cortex-A53
GPU	Dual Core VideoCore
RAM	1GB LPDDR2 SDRAM
Ethernet	Un jack 10/100 RJ45
USB	Cuatro conectores USB 2.0
Memoria	Micro SD card slot

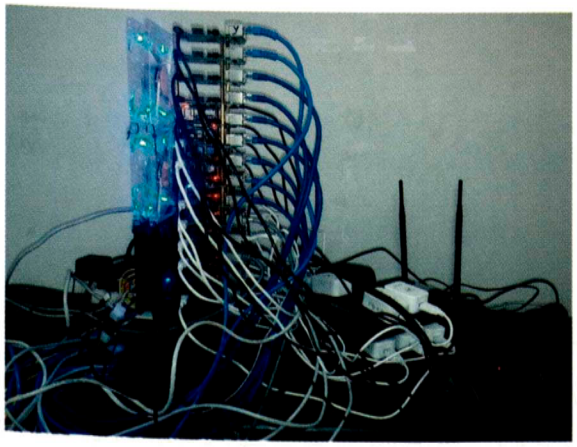

Figure 2. *Clúster Super PI de 116 núcleos con Raspberry Pi2 y Raspberry Pi3.*

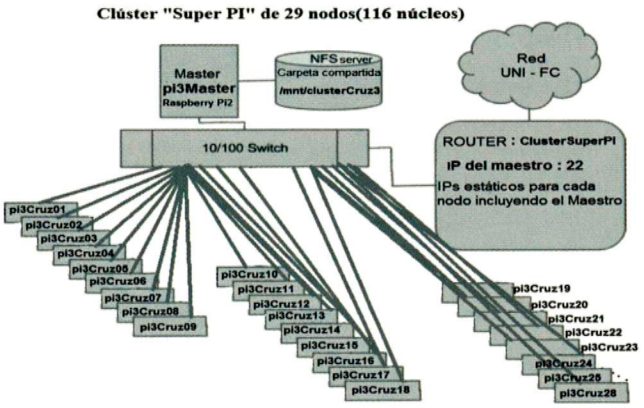

Figure 3. *Arquitectura del Clúster Super PI de 116 núcleos.*

Figure 4. *Ventilador usado en el clúster.*

2.2 Software

Se instaló como sistema operativo Raspbian Jessie Lite, que es una versión optimizada de la distribución Debian GNU/Linux para la Raspberry pi y se trata de una versión aligerada en comparación con Raspbian Jessie. Para ello se preparó una tarjeta MicroSD(figura 5) con una imagen del Raspbian Jessie Lite, descargada del sitio web de la Fundación Raspberry pi(raspberrypi.org).

Figure 5. *Tarjeta de memoria microSD usada en el Clúster Super PI.*

2.2.1 Instalación y Configuración del clúster Súper PI

La secuencia de instalación e configuración del MPICH3, MPI4PY y NFS SERVER se detallan en el Apéndice.

2.2.2 Overclock realizado a los nodos del clúster Súper PI

El overclock nos ha permitido elevar la potencia de computo del clúster. Pero para hacer overclock es necesario disponer de una buena refrigeración(ventiladores) así como de fuentes de alimentación apropiadas para cada nodo como son 5v a 3A(Ver Figura 6) Para tener overclock, esto es una frecuencia de reloj que se eleva a 1,35GHz se modifica el archivo /boot/config.txt (de [5, 6]) de cada nodo, ingresando las líneas:

```
arm_freq=1350
over_voltage=5
sdram_freq=500
gpu_freq=500
```

Figure 6. *Adaptador utilizado como fuente de alimentación para cada nodo en el Clúster Súper PI.*

2.2.3 Temperatura máxima de cada nodo obtenida usando "stress"

El comando "stress" no viene por defecto instalado en Raspbian. Es necesario instalar:

```
sudo apt-get install stress
```

Ejecutamos en cada nodo el comando stress :

```
stress --cpu 4 &
```

luego, ingresamos la siguiente línea de comandos para observar la temperatura y la frecuencia:

```
watch '(vcgencmd measure_temp;vcgencmd \\
measure_clock arm)'
```

El resultado se observa en la siguiente figura :

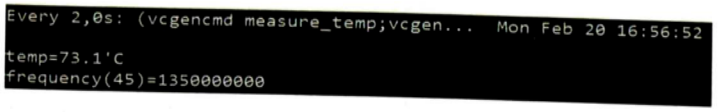

Figure 7. *Medida de la temperatura y frecuencia de un nodo cualquiera en plena ejecución del comando stress .*

2.2.4 Ejecución de programas que calculan el valor de Pi y números primos.

Realizado el overclock para cada nodo, entonces se ejecutan en el clúster Súper PI el programa cpi3, icpi ambos para 800 millones de intervalos y mpiprime1000m que calcula la cantidad de primos encontrados hasta 1000 millones, así como el primo más grande. Ver listado fuente de estos programas cpi3, icpi y mpiprime1000m en el Apéndice.

Resultados obtenidos al ejecutar estos programas: (Ver Figuras 8, 9 y 10)

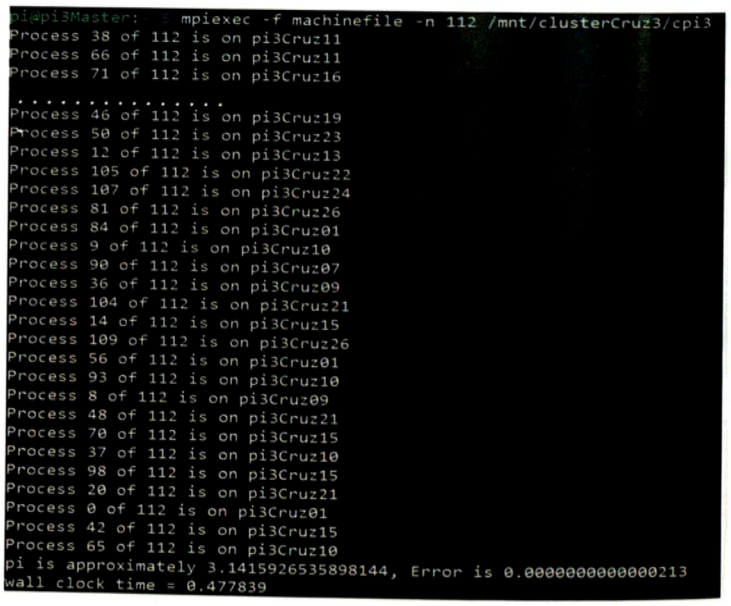

Figure 8. *Ejecución del programa cálculo de Pi(cpi3.c) en el Clúster Súper PI.*

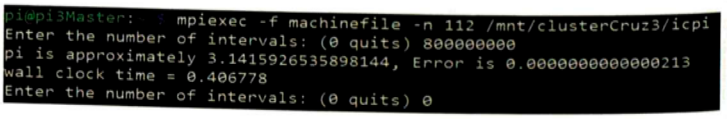

Figure 9. *Ejecución del programa cálculo de Pi interactivo(icpi.c) en el Clúster Súper PI.*

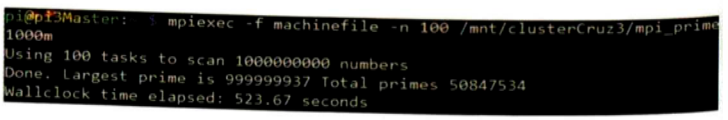

Figure 10. *Ejecución del programa cálculo de números primos hasta 1000 millones(mpiprime1000m.c) en el Clúster Súper PI(se usarón 100 núcleos).*

3 Optimización del consumo de energía en el clúster Super PI

Con el objetivo de bajar el consumo de energía en el clúster, se llevaron a cabo varias acciones que se detallan a continuación.

3.1 Deshabilitando HDMI

Cuando se usa un Raspberry Pi sin monitor, no hay necesidad de encender el circuito del display y se puede ahorrar energía (de [7]), ejecutando :

```
/usr/bin/tvservice -o
```

Así mismo añadiendo la línea a /etc/rc.local se deshabilita HDMI al arrancar.

3.2 deshabilitando el LED ACT y el LED PWR

Se puede ahorrar energía deshabilitando también el LED ACT y el LED PWR. Esto se logra si modifica el archivo /boot/config.txt (de [8]), ingresando las siguientes líneas:

```
dtparam=act_led_trigger=none
dtparam=act_led_activelow=on

dtparam=pwr_led_trigger=none
dtparam=pwr_led_activelow=off
```

3.3 Se deshabilitan el Wifi y Bluetooth

En el clúster que hemos construido no se utilizan las funciones de Wifi y Bluetooth, entonces, para deshabilitar completamente el Wifi y Bluetooth(de [9]) se hace deshabilitando módulos del kernel en Raspbian / OSMC y añadiendo en /etc/modprobe.d/raspi-blacklist.conf las siguientes líneas:

```
#wifi
blacklist brcmfmac
blacklist brcmutil

#bt
blacklist btbcm
blacklist hci_uart
```

4 Comparaciones de tiempo de ejecución entre el Clúster Súper PI y otras Máquinas

En esta sección compararemos los tiempos calculados de programas al ejecutarlos en diferentes computadoras y clústeres. Especificaciones de las máquinas utilizadas en las comparaciones.

4.1 Especificaciones de la computadora PC y clústeres

En la siguiente tabla se listan estas especificaciones.

Tabla 2. Especificaciones de máquinas usadas para las comparaciones.

Máquina	Núcleos	Procesador	RAM
PC	8	i7-4770 3.40GHz	8 GB
clúster CTIC	54	Core2 duo 2.3GHz	27GB
clúster Odroid	80	ARM-2000MHz	40GB
clúster Súper PI	112	ARM-1350MHz	29GB

4.2 Tabla de tiempos de ejecución

La tabla 3 muestra los tiempos obtenidos para cada máquina.

Tabla 3. Tiempos calculados al ejecutar diversos programas para cada máquina.

Máquina	Programa	Tiempo
PC	cpi3	0,774s
PC	icpi	0,763s
PC	mpiprime1000m	357.54s
clúster CTIC	cpi3	0,563s
clúster Súper PI	cpi3	0.477s
clúster Súper PI	icpi	0.406s
clúster Súper PI	mpiprime1000m	523.67s
clúster Odroid	cpi3	0.255s
clúster Odroid	icpi	0.248s
clúster Odroid	mpiprime1000m	473.61s

En las siguientes gráficas(figuras 11, 12 y 13) podemos notar mejor las comparaciones entre las máquinas. Dependiendo de los programas ejecutados una máquina supera a otra. El clúster ODROID supera a los demás en el tiempo de ejecución de los programas CPI3 y ICPI. El clúster SUPER PI supera a la PC y al clúster CTIC en los tiempos de ejecución de los programas CPI3 y ICPI. La PC supera al clúster ODROID y el clúster Súper PI en el tiempo de ejecución del programa cálculo de números primos hasta un límite de 1000 millones.

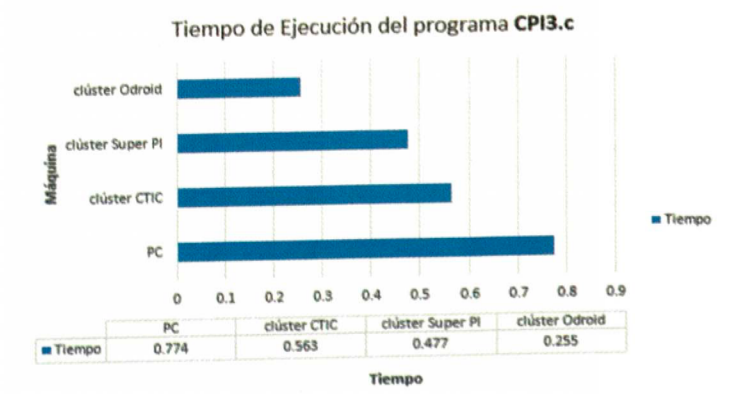

Figure 11. *Gráfica de tiempos de ejecución del programa cálculo de Pi(cpi3.c) en diferentes máquinas(Menor tiempo es mejor performance).*

Figure 12. *Gráfica de tiempos de ejecución del programa cálculo de Pi interactivo(icpi.c) en diferentes máquinas(Menor tiempo es mejor performance)*

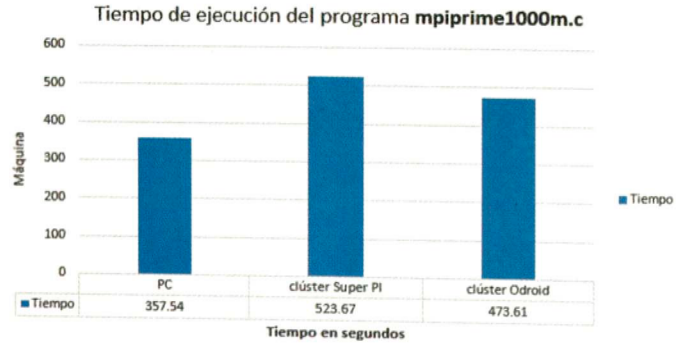

Figure 13. *Ejecución del programa de cálculo de números primos hasta 1000 millones(mpiprime1000m.c) en diferentes máquinas(Menor tiempo es mejor performance).*

5 Conclusiones

Concluimos que con placas Raspberry Pi3 ha sido posible construir un clúster de una potencia de computo superior al clúster de PCs recicladas del año 2008(clúster CTIC). El clúster Súper PI frente al clúster CTIC tiene como otras ventajas de ser de menor consumo de energía y de menor costo en su mantenimiento. El clúster Súper PI ocupa un espacio reducido, una quinta parte de lo que ocupa el clúster CTIC. El funcionamiento de los nodos es estable en comparación con los nodos del clúster CTIC. El uso de minicomputadores Raspberry Pi3 basados en procesadores ARM para construir un clúster es una alternativa válida al uso de los computadores clásicos basados en procesadores INTEL o AMD. Este hecho es igualmente corroborado por el clúster ODROID que ha sido construido con 20 placas ODROID XU4 y tiene una potencia de computo nada despreciable. Si una ventaja podemos nombrar del clúster Súper PI frente al clúster ODROID es que consume menos energía. En un futuro, con el desarrollo tecnológico aparecerán procesadores basados en ARM más potentes que harán que se recorte cada vez más la brecha en potencia de computo frente a los procesadores clásicos INTEL o AMD. Los resultados obtenidos nos animan a seguir escalando el clúster Super PI y seguir usando procesadores basados en ARM, hasta conseguir una verdadera SUPER COMPUTADORA !!.

6 Trabajo Futuro

Como una continuidad de este trabajo se usarán en un futuro cercano para construir un clúster placas PINE64 que son superiores en rendimiento a las placas Raspberry pi 3.

7 Agradecimientos

Agradecemos a Dios, a la UNI y a la Facultad de Ciencias (Of. R2-313).

Apéndice

Instalación y configuración de MPICH3, MPI4PY y NFS SERVER [10, 11]

```
Una vez grabado la imagen del sistema operativo \
Raspbian Jessie Lite en el microSD insertamos \
este en el raspberry pi. En el terminal, \
ingrese sudo raspi-config. Aquí, expandimos el \
sistema de archivos, cambiamos de acuerdo al \
idioma y país las opciones de internalización,\
habilitamos ssh, cambiamos la memoria GPU a \
16MB, seleccionamos arranque en consola \
automáticamente logueado como usuario pi,\
cambiamos hostname a pi3Master \
y finalmente reiniciamos. Actualización\
de Linux
sudo apt-get update
mkdir mpich3
wget http://www.mpich.org/static/downloads \
/3.2/mpich-3.2.tar.gz
cd mpich3
cp /home/pi/mpich-3.2.tar.gz /home/pi/mpich3
tar xfz mpich-3.2.tar.gz
sudo mkdir /home/rpimpi
sudo mkdir /home/rpimpi/mpi-install
mkdir /home/pi/mpi-build
cd /home/pi/mpi_build
Instalación de Fortran
sudo apt-get install gfortran
sudo make
sudo make install
cd
sudo nano .bashrc
Añade ruta para mpi dentro de este archivo
PATH=$PATH:/home/rpimpi/mpi-install/bin
sudo reboot
Prueba si mpi está trabajando en este nodo
mpiexec -n 1 hostname
Instalación librerías de desarrollo en Python
```

```
sudo aptitude install  Python-dev
Instalación mpi4py
Usar wget para descargar archivo \
mpi4py-1.3.1.tar.gz
wget https://mpi4py.googlecode.com/files \
/mpi4py-1.3.1.tar.gz
tar -zxf mpi4py-1.3.1.tar.gz
cd mpi4py-1.3.1/
python setup.py build
sudo python setup.py install
Fijamos ruta de ambiente
export PYTHONPATH=/home/pi/mpi4py-1.3.1
Prueba mpi4py sobre el nodo 1
mpiexec -n 4 python demo/helloworld.py
Removemos el microSD del nodo 1 e insertamos \
en la laptop para hacer una copia de esta \
imagen que se usará para el nodo 2.
Usamos Win32DiskImager para leer la imagen \
hacia la PC.
Regresamos el microSD al nodo1.
Grabamos la imagen a un nuevo microSD que \
se insertará en el nodo 2.
Asignamos IPs estáticos
Para ello ingresamos a interfaces y escribimos \
 la dirección IP para cada nodo
Nodo 1: 192.168.0.140
Nodo 2: 192.168.0.141
sudo nano /etc/network/interfaces
Luego, reiniciar el nodo correspondiente
sudo reboot
Cambiamos el hostname del nodo 2 a pi3Cruz01
sudo raspi-config
Reiniciamos
Cambiamos los nombres de hosts del nodo 1 y\
 el nodo 2. Añadimos
los IPs estáticos
sudo nano /etc/hosts
El nodo maestro necesita automáticamente \
autenticarse. Añadimos algunas llaves.
En el nodo maestro
ssh-keygen
cd .ssh
cp id_res_pub pi3Master
En el nodo 2
ssh pi3Cruz01
ssh-keygen
cd .ssh
scp 192.168.0.140:/home/pi/.ssh/pi3Master .
cat pi3Master>>authorized-keys
exit
Ahora, el ssh al nodo 2 debería \
de trabajar sin password
ssh pi3Cruz01
Clonamos el microSD del nodo pi3Cruz01\
 para los demás nodos.
Instalación del NFS Server
Necesitamos una carpeta compartida que todos \
los nodos puedan accesar.
En el nodo 1(pi3Master) hacemos:
sudo apt-get install nfs-kernel-server
sudo mkdir /mnt/clusterCruz3
Edita el archivo exports para añadir una ruta a \
la carpeta compartida
sudo nano /etc/exports
Incluir al final:
/mnt/clusterCruz3 *(rw,sync,no_root_squash, \
no_subtree_check)
Modificar acceso a carpeta compartida
sudo chmod -R 777 /mnt/clusterCruz3/
Aplica cambios al nfs server
sudo update-rc.d rpcbind enable
sudo service rpcbind restart
sudo exportfs -a
sudo service nfs-kernel-server restart
Sobre el nodo 2(pi3Cruz01) se monta la \
carpeta compartida del nodo 1(pi3Master)
Ir al nodo 2:
ssh pi3Cruz01
sudo mkdir /mnt/clusterCruz3
sudo mount pi3Master:/mnt/clusterCruz3 \
/mnt/clusterCruz3/
df -h
Importante:
Evitamos el automontaje de la carpeta \
compartida porque reduce el rendimiento \
de cada nodo a la mitad.
Por eso motivo no se necesita editar el \
archivo fstab.

Modificación del clone de la imagen \
pi3Cruz01 para obtener otros nodos \
 como por ejemplo pi3Cruz21
En el master, añado un nuevo nodo 192.168.0.161\
pi3Cruz21 en hosts
pi@pi3Master:~ $ sudo nano /etc/hosts
Aquí también modifico el archivo machinefile \
 añadiendo una nueva línea a pi3Cruz21
pi@pi3Master:~ $ nano machinefile
Vamos al nodo pi3Cruz01 que es la que vamos \
a convertir en el nodo pi3Cruz21
pi@pi3Master:~ $ ssh pi3Cruz01
En el nodo ?pi3Cruz01? realizo la modificación \
del IP para tener 192.168.0.161
pi@pi3Cruz01:~ $ sudo nano /etc/network/interface
Añado el IP en el archivo hosts
pi@pi3Cruz01:~ $ sudo nano /etc/hosts
Ingreso al archivo fstab
pi@pi3Cruz01:~ $ sudo nano /etc/fstab
En fstab comento la línea pi3Master:/mnt \
/clusterCruz3 /mnt/clusterCruz3 nfs rw 0 0
Ejecuto raspi-config
pi@pi3Cruz01:~ $ sudo raspi-config
Selecciono ?opciones avanzadas?

Es hora de probar el clúster
Chequeamos si todos los nodos están \
activos (probamos para 4 nodos)
En el maestro:
mpiexec -n 4 -hosts pi3Cruz01,pi3Cruz02,\
pi3Cruz03,pi3Cruz04 hostname
pi3Cruz04
pi3Cruz02
```

```
pi3Cruz03
pi3Cruz01
o ejecutamos:
mpiexec - f machinefile  -n 4  hostname
pi3Cruz04
pi3Cruz02
pi3Cruz01
pi3Cruz03
Ahora probamos MPI4Py
mpiexec -hosts pi3Cruz01,pi3Cruz02,pi3Cruz03,\
pi3Cruz04 -n 16 python /mnt/clusterCruz3\
/helloworld.py
Hello, World! I am process 0 of 16 on pi3Cruz01.
Hello, World! I am process 2 of 16 on pi3Cruz03.
Hello, World! I am process 4 of 16 on pi3Cruz01.
Hello, World! I am process 6 of 16 on pi3Cruz03.
Hello, World! I am process 8 of 16 on pi3Cruz01.
Hello, World! I am process 10 of 16 on pi3Cruz03.
Hello, World! I am process 12 of 16 on pi3Cruz01.
Hello, World! I am process 14 of 16 on pi3Cruz03.
Hello, World! I am process 5 of 16 on pi3Cruz02.
Hello, World! I am process 9 of 16 on pi3Cruz02.
Hello, World! I am process 1 of 16 on pi3Cruz02.
Hello, World! I am process 13 of 16 on pi3Cruz02.
Hello, World! I am process 7 of 16 on pi3Cruz04.
Hello, World! I am process 15 of 16 on pi3Cruz04.
Hello, World! I am process 11 of 16 on pi3Cruz04.
Hello, World! I am process 3 of 16 on pi3Cruz04.
```

Programa cpi3.c

```
/* -*- Mode: C; c-basic-offset:4 ;  \
indent-tabs-mode:nil ; -*- */
/*
*  (C) 2001 by Argonne National Laboratory.
*/
#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f(double);

double f(double a)
{
return (4.0 / (1.0 + a*a));
}

int main(int argc,char *argv[])
{
int    n, myid, numprocs, i;
double PI25DT = 3.14159265358979323846264643;
double mypi, pi, h, sum, x;
double startwtime = 0.0, endwtime;
int    namelen;
{
return (4.0 / (1.0 + a*a));
}

int main(int argc,char *argv[])
{
int    n, myid, numprocs, i;
double PI25DT = 3.14159265358979323846264643;
double mypi, pi, h, sum, x;
double startwtime = 0.0, endwtime;
int    namelen;
char processor_name[MPI_MAX_PROCESSOR_NAME];

MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
MPI_Get_processor_name(processor_name,&namelen);

fprintf(stdout,"Process %d of %d is on %s\n",
myid, numprocs, processor_name);
fflush(stdout);
n = 800000000; /* # de intervalos modificado */
if (myid == 0)
   startwtime = MPI_Wtime();
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
h   = 1.0 / (double) n;
sum = 0.0;
/* A slightly better approach starts from \
large i and works back */
for (i = myid + 1; i <= n; i += numprocs)
{
x = h * ((double)i - 0.5);
sum += f(x);
}
mypi = h * sum;
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, \
MPI_SUM, 0, MPI_COMM_WORLD);

if (myid == 0) {
endwtime = MPI_Wtime();
printf("pi is approximately %.16f, Error\
 is %.16f\n",
pi, fabs(pi - PI25DT));
printf("wall clock time = %f\n", \
endwtime-startwtime);
fflush(stdout);
}
 MPI_Finalize();
 return 0;
}
```

Programa icpi.c

```
/* -*- Mode: C; c-basic-offset:4 ; \
indent-tabs-mode:nil ; -*- */
/*
*  (C) 2001 by Argonne National Laboratory.
*/

/* This is an interactive version of cpi */
#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f(double);

double f(double a)
{
```

```
return (4.0 / (1.0 + a*a));
}

int main(int argc,char *argv[])
{
int done = 0, n, myid, numprocs, i;
double PI25DT = 3.14159265358979323846264 3;
double mypi, pi, h, sum, x;
double startwtime = 0.0, endwtime;
int  namelen;
char processor_name[MPI_MAX_PROCESSOR_NAME];
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
MPI_Get_processor_name(processor_name,&namelen);
while (!done) {
 if (myid == 0) {
 fprintf(stdout, "Enter the number of intervals: \
 (0 quits) ");
 fflush(stdout);
 if (scanf("%d",&n) != 1) {
 fprintf( stdout, "No number entered; quitting\n" );
 n = 0;
 }
 startwtime = MPI_Wtime();
 }
 MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
 if (n == 0)
 done = 1;
 else {
 h   = 1.0 / (double) n;
 sum = 0.0;
 for (i = myid + 1; i <= n; i += numprocs) {
 x = h * ((double)i - 0.5);
 sum += f(x);
 }
 mypi = h * sum;
 MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, \
 MPI_SUM, 0, MPI_COMM_WORLD);

 if (myid == 0) {
 printf("pi is approximately %.16f, \
 Error is %.16f\n",
 pi, fabs(pi - PI25DT));
 endwtime = MPI_Wtime();
 printf("wall clock time = %f\n", \
 endwtime-startwtime);
 fflush( stdout );
  }
 }
}
 MPI_Finalize();
 return 0;
}
```

Programa mpiprime1000m.c

```
/******************************************
* FILE: mpiprime1000m.c
* DESCRIPTION:
*   Generates prime numbers.
* AUTHOR: Blaise Barney 11/25/95 - adapted from\
* version contributed by Richard Ng &  Wong Sze\
* Cheong during MHPCC Singapore Workshop (8/22/95).
* LAST REVISED: 04/13/05
******************************************/
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define LIMIT  1000000000  /* Increase this\
 to find more primes */
#define FIRST  0         /* Rank of first task */

int isprime(int n) {
int i,squareroot;
if (n>10) {
squareroot = (int) sqrt(n);
 for (i=3; i<=squareroot; i=i+2)
 if ((n%i)==0)
 return 0;
 return 1;
 }
 /* Assume first four primes are counted \
 elsewhere. Forget everything else */
 else
 return 0;
 }

 int main (int argc, char *argv[])
 {
 int   ntasks,       /* total number of \
 tasks in partitiion */
 rank,               /* task identifier */
 n,                  /* loop variable */
 pc,                 /* prime counter */
 pcsum,              /* number of primes \
 found by all tasks */
 foundone,           /* most recent \
 prime found */
 maxprime,           /* largest prime found */
 mystart,            /* where to start \
 calculating */
 stride;             /* calculate every nth\
  number */

 double start_time,end_time;
 MPI_Init(&argc,&argv);
 MPI_Comm_rank(MPI_COMM_WORLD,&rank);
 MPI_Comm_size(MPI_COMM_WORLD,&ntasks);
 if (((ntasks%2) !=0) || ((LIMIT%ntasks) !=0)) {
 printf("Sorry - this exercise requires \
 an even number of tasks.\n");
 printf("evenly divisible into %d.  Try 4 or \
 8.\n",LIMIT);
 MPI_Finalize();
  exit(0);
  }

  start_time = MPI_Wtime();  /* Initialize \
```

```
start time */
mystart = (rank*2)+1;       /* Find my \
starting point\
 - must be odd number */
stride = ntasks*2;          /* Determine stride, \
skipping even numbers */
pc=0;                       /* Initialize \
prime counter */
foundone = 0;               /* Initialize */

/*** task with rank 0 does this part ***/
if (rank == FIRST) {
printf("Using %d tasks to scan %d numbers\n"\
,ntasks,LIMIT);
pc = 4;                     /* Assume first\
 four primes are \
counted here */
for (n=mystart; n<=LIMIT; n=n+stride) {
if (isprime(n)) {
pc++;
foundone = n;
/***** Optional: print each prime \
as it is found
printf("%d\n",foundone);
*****/
}
}
MPI_Reduce(&pc,&pcsum,1,MPI_INT,MPI_SUM,FIRST,\
 MPI_COMM_WORLD);
 MPI_Reduce(&foundone,&maxprime,1,MPI_INT,MPI_
 FIRST,MPI_COMM_WORLD);
 end_time=MPI_Wtime();
 printf("Done. Largest prime is %d Total prime
 %d\n",maxprime,pcsum);
 printf("Wallclock time elapsed: %.2lf seconds
 end_time-start_time);
 }
/*** all other tasks do this part ***/
if (rank > FIRST) {
 for (n=mystart; n<=LIMIT; n=n+stride) {
 if (isprime(n)) {
 pc++;
 foundone = n;
 /***** Optional: print each prime as it is fo
 printf("%d\n",foundone);
 *****/
 }
 }
 MPI_Reduce(&pc,&pcsum,1,MPI_INT,MPI_SUM,FIRST,
 MPI_COMM_WORLD);
 MPI_Reduce(&foundone,&maxprime,1,MPI_INT,\
 MPI_MAX,FIRST,MPI_COMM_WORLD);
 }
 MPI_Finalize();
 }
```


1. Cruz C. M., REVCIUNI, 16(1),1–6, 6 (2013)
2. Cruz, C. M.,REVCIUNI, 17(1),9–16, 8 (2014)
3. Cruz C. M., REVCIUNI, 17(1), 50–57, 10 (2014)
4. ODROID XU4. https://www.hardkernel.com/main/products/prdtinfo.php?gcode=G143452239825 (2016)
5. Overclocking the Raspberry PI3. https://jonlennartaasenden.wordpress.com/2016/11/17/overclocking-the-raspberry-pi-3/ , November 17 (2016)
6. James H., Raspberry Pi 3 Overclock and Turbo Config https://haydenjames.io/raspberry-pi-3-overclock, May 13 (2016)
7. Geerling J., Raspberry Pi Zero. Conserve power and reduce draw to 80mA https://www.jeffgeerling.com/blogs/jeff-geerling/raspberry-pi-zero-conserve-energy/ , November 29 (2015)
8. Geerling J., Controlling PWR and ACT LEDs on the Raspberry Pi. https://www.jeffgeerling.com/blogs/jeff-geerling/controlling-pwr-act-leds-raspberry-pi, March 15th (2015)
9. Raspberry Pi3 disable bluetooth and wifi. http://pingtool.org/raspberry-pi-3-disable-bluetooth-wifi/, June 14th, (2016)
10. Cluster computing with three Raspberry Pi 3 boards https://www.raspberrypi.org/magpi/cluster-comput -raspberry-pi-3/ , October (2016)
11. Raspberry Pi 3 Cluster Supercomputer http://www.rasmurtech.com/raspberry-pi-3-cluster-supercomputer/ (2016)