Medidas de rendimiento y comparación entre el Clúster Cruz I y el Clúster Cruz II

César Martín Cruz Salazar†
CTIC(Centro de Tecnologias de Información y Comunicaciones) - Facultad de Ciencias. Universidad Nacional de Ingeniería;
†*ccruz@uni.edu.pe*

Recibido el 04 de Agosto del 2014; aceptado el 15 de Setiembre del 2014

Este artículo trata sobre medidas de rendimiento(Benchmarking) efectuadas en un clúster y las comparaciones de los tiempo de ejecución que le toma al clúster CruzI ejecutar ciertos programas(segun [1] las siglas CRUZ significan Clúster de Investigación de la Uni de potencia Zero) que contiene 16 nodos esclavos y un nodo adicional para el nodo maestro con los tiempos que toma al Clúster CruzII de 6 nodos esclavos y un nodo para el maestro. Se ejecutó el mismo programa de cálculo de PI de forma paralela utilizando funciones MPI en ambas máquinas y se midió el tiempo que toma este calculo. Así también se ejecutó el software hpl (linpack de alto rendimiento) para obtener la velocidad de computo del clúster CruzII en unidades de gigaflops.
Palabras Claves: clúster, raspberry pi versión 2, programación paralela, programación distribuida, MPI, clúster educativo, portátil, comparación.

This paper discusses performance measures (Benchmarking) made in a cluster and comparisons of run time it takes the cluster CruzI to run certain programs (according to [1] CRUZ means Cluster Research Uni of Zero Power) containing 16 slave nodes and an additional node to the master node with the time it takes the Cluster CruzII 6 slave nodes and one master node. The same PI calculation program executed in parallel using MPI functions on both machines and the time it takes this calculation was measured. The software HPL (High Performance Linpack) was also executed for computing speed CruzII cluster units gigaflops.
Keywords: cluster, raspberry pi versio 2, parallel computing, distributed computing, MPI, education cluster, portable, comparation.

1 Introducción

Motivados a continuar siempre utilizando el Raspberry Pi (RPI) en la escuela de Ciencia de la Computación de la Facultad de Ciencias de la UNI es que construimos el clúster Cruz II para promover conceptos mas avanzados de programación paralela y computación de alto rendimiento. Medimos su rendimiento en gigaflops usando programas comunes para estos casos, los mismos que se utilizan para clasificar las supercomputadoras de acuerdo con su rendimiento en la clasificatoria Top500. Luego, lo comparamos con el clúster Cruz I que construimos anteriormente [1] para observar el aumento de potencia de computo en su verdadera dimensión que obtenemos con el lanzamiento de la versión 2 del RPI.Es así que en este artículo presentamos los resultados de las mediciones y de las comparaciones entre el clúster "Cruz I" y el clúster "Cruz II". El Clúster Cruz II fue construido usando 6 placas Raspberry Pi 2 y 1 placa Raspberry Pi B+ interconectadas con enlaces Ethernet de 100 Mbtis/s. Un rack que comprende las 7 placas RPIs que van conectados a switches se muestra en la Figura 1. El bajo costo del clúster combinado por su peso ligero y tamaño compacto, lo hacen adecuado para la educación y una serie de aplicaciones que demanda la programación paralela y que un clúster convencional lo ejecuta muy bien pero que debido a sus altos costos y requisitos especiales de infraestructura su uso para la enseñanza resulta poco adecuado.

1.1 Contexto y trabajo relacionado

Considerar el bajo consumo de energía es la tendencia para este tipo de desarrollos. La computación intensiva de datos está llegando a ser una área de gran interés, tanto en el ámbito académico como en la industria. Aplicaciones ricas en datos son cada vez más frecuentes.

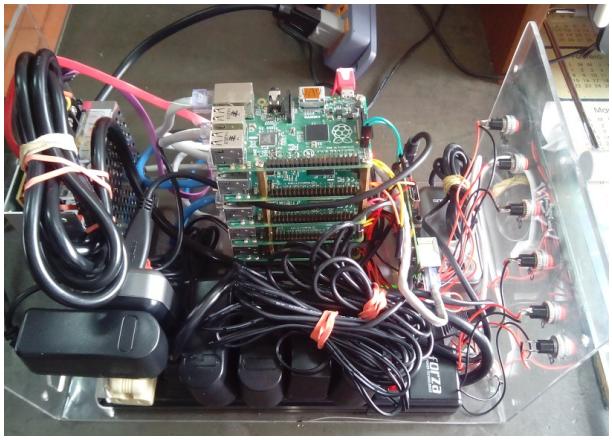

Figura 1. *Una pila(rack) del Clúster CruzII conectados a switches*

2 El presente trabajo

El Clúster mostrado en la Figura 2 que utilizamos en este trabajo se destaca por el uso de procesadores ARM de bajo costo y de bajo consumo de energía, conectados en red usando Ethernet y switches. La arquitectura de este Clúster se muestra en la Figura 3 y está basada en el primer trabajo realizado por el autor [1] donde se muestra un clúster de 16 nodos más un nodo maestro. Para la construcción de nuestro Clúster se dispone del apoyo tradicional de tecnologías como MPI (Message Passing Interface), sobre la cual muchas aplicaciones de supercomputación son desarrolladas. Se dan en la Sección 3 las especificaciones del Clúster, describiendo su hardware y software. En la Sección 4 dirigimos nuestra atención a la medición del rendimiento del clúster utilizando HPL. En la Sección 5 hacemos comparaciones entre el clúster Cruz I y el clúster Cruz II mediante la ejecución de programas en paralelo. Y en la Sección 6 se hacen comparaciones entre una PC y el clúster Cruz II.

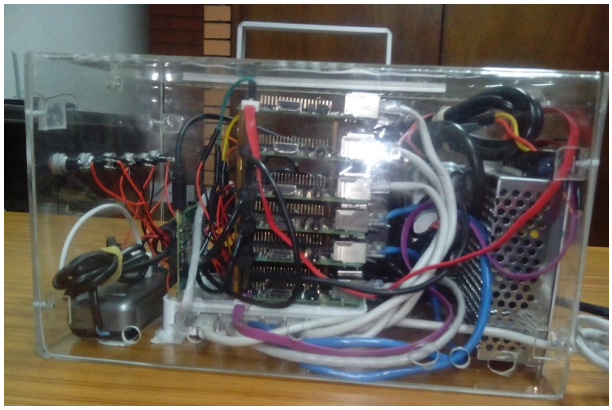

Figura 2. *El Clúster "CruzII"*

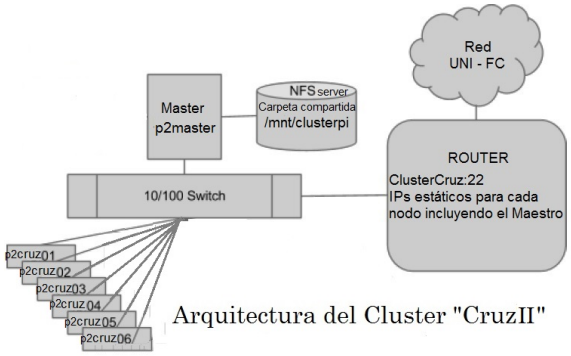

Figura 3. *Arquitectura del Clúster "CruzII"*

3 Descripción del sistema

Proporcionamos una descripción de la arquitectura del clúster, tanto en términos de sus componentes de hardware y de su entorno de software.

3.1 Hardware

El Clúster "Cruz II" se compone de 6 Raspberry Pi versión 2 que actúan como nodos esclavos más una placa Raspberry Pi modelo B+ que actúa como maestro. Estas placas son del tamaño de una tarjeta de crédito. Cada uno cuenta con 1 Gigabyte de RAM, un sistema sobre un Chip(SoC, Sistema sobre un Chip) Broadcom BCM2836, que integra un procesador quad core Arm7 de 900MHz, un GPU Broadcom VideoCore IV con salida HDMI y salida de vídeo RCA, y un controlador USB. La placa también contiene, un adaptador para puerto Ethernet externo de 10/100 Mbits, y un adaptador de cuatro puertos USB 2.0. El almacenamiento local se realiza a través de una tarjeta de memoria microSD de 8 gigabytes que se coloca en una ranura acondicionada para esta tarjeta. Cada placa RPI2 se alimenta con una fuente de 5 voltios que son suministrados por adaptadores de voltaje. Cada placa RPI2 utilizada en el clúster se encuentra overclockeado a 1 Ghz.

3.2 Software

Instalamos el sistema operativo Raspbian, que es una versión optimizada de la distribución Debian GNU/Linux para la Raspberry Pi. Para ello hemos preparado una tarjeta MicroSD con una imagen descargada de la Fundación Raspberry Pi(raspberrypi.org), y personalizamos esta instalación añadiendo software necesario para ejecutar programas en paralelo(MPI y NFS server). Esta viene a ser la imagen del "maestro". Luego, hemos clonado esta imagen "maestro", a las tarjetas microSD de los demás nodos. Para configurar el Clúster se siguieron los pasos mencionados en la referencia [2], los indicados en la referencia [4] y [3].

3.2.1 Linpack

Linpack[5] es un programa implementado ya sea en Fortran o C para realizar cálculos de álgebra lineal numérica en computadoras. Fue desarrollado inicialmente como un benchmark[6] para medir el número flops en supercomputadoras. Linpack resuelve un denso sistema NxN de ecuaciones lineales Ax = b, ya que es un problema común en la ingeniería y la computación científica. Linpack hace

uso de la Biblioteca de Álgebra Lineal Básica (BLAS) para realizar operaciones con matrices y vectores.

3.2.2 HPL(High Performance Linpack)

Linpack de Alto Rendimiento (HPL) es una implementación paralela del Linpack benchmark y es portátil en un amplio número de máquinas. HPL utiliza aritmética de 64 bits de doble precisión para resolver un sistema de ecuaciones lineales de orden N[7] . Generalmente se ejecuta en las computadoras de memoria distribuida para determinar el rendimiento de punto flotante de doble precisión del sistema. Utiliza MPI para la comunicación entre nodos y se basa en varias rutinas de la biblioteca BLAS. El archivo de entrada, HPL.dat del HPL benchmark ofrece información sobre el tamaño del problema, el tamaño del bloque, la dimensión del grilla, etc. Este archivo de entrada puede ser ajustado para un rendimiento de acuerdo al sistema en el que HPL benchmark es ejecutado y de la topología de la red utilizada para interconectar los nodos.

3.2.3 Benchmarking con High Performance Linpack(Técnica de medida de rendimiento en GigaFlops usando HPL)

Para instalar e configurar HPL hemos seguido la referencia [8] que cubre cómo hacer la evaluación comparativa(benchmarking) de un sistema con Raspberry Pi 2. Primero haremos el benchmark de un solo nodo, y luego continuamos a hacer el benchmark de varios nodos, cada nodo representa un Raspberry Pi. Hay algunas cosas que deben observarse aquí. En primer lugar, la evaluación comparativa de un solo nodo o varios nodos para que funcione tienen que tener instalado algunas dependencias. Esto entre otros, es por ejemplo la implementación de MPI (como MPICH o OpenMPI) que debe estar instalado y en funcionamiento para que HPL pueda trabajar. Así que para la evaluación comparativa de varios nodos, se asume que todos los nodos tienen MPICH instalado y funcionando. Para instalar otras dependencias y paquetes, utilizamos el siguiente comando:

```
pi@p2master ~ $ sudo apt-get install \
libatlas-base-dev libmpich2-dev gfortran
```

Sólo este paso tiene que ser repetido en cada uno de los nodos (PiS) presentes en el clúster. Descargamos el paquete HPL del sitio web indicado en [8] y ejecutamos el siguiente conjunto de comandos uno después de otro en el terminal de comandos:

```
tar xf hpl-2.1.tar.gz
mv hpl-2.1 hpl
cd hpl/setup
sh make_generic
cd ..
cp setup/Make.UNKNOWN Make.rpi
```

Abrimos el archivo Make.rpi usando el comando :

```
pi@p2master ~ $ nano Make.rpi
```

Y hacemos los siguientes cambios al archivo :

```
ARCH         =rpi
TOPdir       =$(HOME)/hpl
MPdir        =
MPinc        =
MPlib        =-lmpi
LAdir        = /usr/lib/atlas-base/
LAlib        = -lblas
```

Una vez que el archivo Make.rpi está listo, podemos empezar con la compilación del HPL. El archivo ".xhpl" aparecerá en la carpeta "bin/rpi" dentro de la carpeta de hpl. Ejecutamos el siguiente comando:

```
pi@p2master ~ $ make arch=rpi
```

Lo que sigue es un ejemplo del archivo "HPL.dat". Este es un archivo de entrada para HPL cuando se ejecuta. Los valores indicados en este archivo se utilizan para generar y calcular el problema. Puede utilizar este archivo directamente para ejecutar pruebas de un solo nodo. Creamos el archivo con nombre HPL.dat en la carpeta "bin/rpi". Para una memoria RAM de 920Mbytes, 6 nodos por 4 núcleos por nodo, se ha calculado el contenido del archivo usando [9] y se da a continuación.

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of\
Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
23808         Ns
1            # of NBs
128          NBs
0            PMAP process mapping
1            # of process grids (P x Q)
4            Ps
6            Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
1            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,
             4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed)
             form
0            U  in (0=transposed,1=no-transposed)
             form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

Luego, copiamos este archivo HPL.dat desde el maestro a cada uno de los demás nodos del clúster:

```
pi@p2master ~ $ scp HPL.dat p2cruz01:/home/pi/
hpl/bin/rpi
pi@p2master ~ $ scp HPL.dat p2cruz02:/home/pi/
hpl/bin/rpi
pi@p2master ~ $ scp HPL.dat p2cruz03:/home/pi/
hpl/bin/rpi
pi@p2master ~ $ scp HPL.dat p2cruz04:/home/pi/
hpl/bin/rpi
pi@p2master ~ $ scp HPL.dat p2cruz05:/home/pi/
hpl/bin/rpi
pi@p2master ~ $ scp HPL.dat p2cruz06:/home/pi/
hpl/bin/rpi
```

4 Resultados del programa de Benchmarking HPL en el Clúster

4.1 Obtención de la velocidad en Gigaflops de un nodo

El contenido del archivo pifile sería:

```
#nodo 1
172.20.45.247
```

Luego, para ejecutar HPL en el terminal de comandos ingresamos para un nodo de 4 núcleos:

```
pi@p2master ~ $ ~/hpl/bin/rpi $ mpiexec -f \
~/mpi_testing /pifile -n 4 ./xhpl
```

El resultado lo puede ver en la Figura 4:

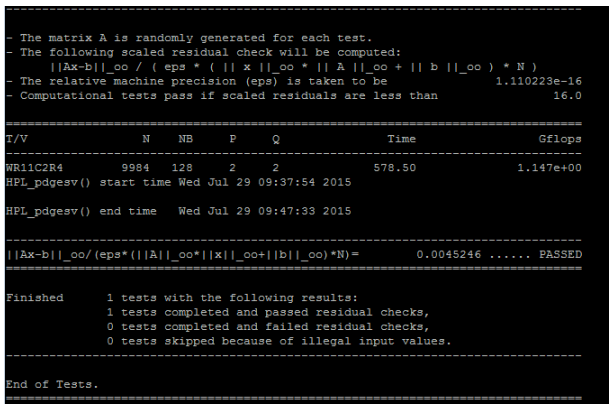

Figura 4. *Resultado de la ejecución del programa HPL en un nodo del clúster "Cruz II"*

Podemos notar que para 1 nodo de 4 núcleos se obtiene una velocidad de 1.147Gigaflops.

4.2 Tabla obtenida luego de varias ejecuciones del programa HPL

El contenido del archivo pifile para este caso sería:

```
#nodo 1
172.20.45.247
#nodo 2
172.20.45.246
#nodo 3
172.20.45.245
#nodo 4
172.20.45.244
#nodo 5
172.20.45.243
#nodo 6
172.20.45.242
```

La tabla de abajo muestra los resultados al ejecutar varias veces la siguiente línea en el terminal de comandos del clúster:

```
pi@p2master ~ $ ~/hpl/bin/rpi $ mpiexec -f \
~/mpi_testing /pifile -n 24 ./xhpl
```

Para un clúster de 6 nodos con 4 núcleos por nodo y tamaño de bloque de 128 :

Memoria Nodo(Mb)	Ns	Tiempo(s)	Gigaflops
450	16640	617	4.979
500	17640	722.84	5.063
550	18432	798.20	5.231
600	19200	900.54	5.240
650	19968	1006.42	5.275
700	20736	1107.03	5.370
750	21504	1227.94	5.399
770	21760	1260.27	5.451
800	22272	1340.47	5.495
850	23040	1482.06	5.502
880	23296	1522.40	5.537
900	23552	1567.57	5.557
920	23808	1610.44	5.587
930	24064	1703.13	5.455
950	24320	1983.28	4.836

Notamos, que el clúster CruzII con 6 nodos con 24 núcleos obtiene una velocidad máxima de 5.587Gigaflops. Valor obtenido considerando un tamaño de memoria RAM de 920Mbytes. Si continuamos aumentando el tamaño de memoria esta velocidad empieza a disminuir. El procedimiento seguido en la toma de medidas se hicieron de acuerdo a [10]

5 Ejecución de programas en los clústeres Cruz I y Cruz II

Los programas que se ejecutarón vinieron como ejemplos al descargar el archivo de instalación del MPI:

```
http://www.mpich.org/static/downloads/3.0.4/
mpich-3.0.4.tar.g
```

5.1 Ejecución del programa cpi2

Entre estos programa se encuentra el "cpi" y en la Figura 5 se muestra la ejecución del programa "cpi2" que calcula el valor de "pi", este programa ha sido modificado para considerar un número de intervalos de 180 millones y el resultado obtenido considerando 16 nodos. El archivo ejecutable se coloca en la carpeta compartida /mnt/clusterpi. Para ejecutar el programa en la línea de comando ingresamos:

```
pi@MasterCruz ~ $mpiexec -f pifile -n 16
/mnt/clusterpi/cpi2
```

Se observa que se demoró en ejecutar un tiempo de 1,723033 segundos.

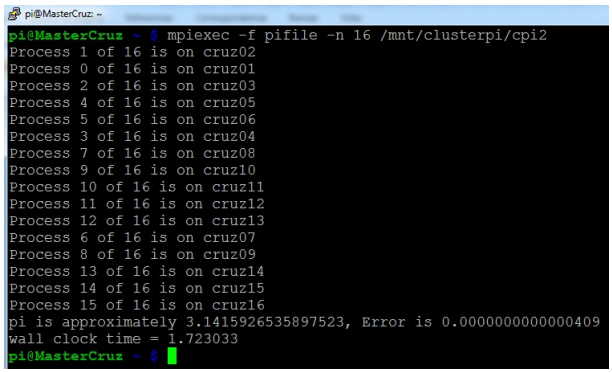

Figura 5. *Ejecución de un programa cpi2 con MPI en el clúster "Cruz I"*

Ejecución del mismo programa en el Clúster Cruz II utilizando 6 nodos con 16 núcleos ver Figura 6:

```
pi@p2master ~ $ mpiexec -f pifile -n 16
/mnt/clusterpi/cpi2
```

Demoró en ejecutar un tiempo de 1,115871 segundos.

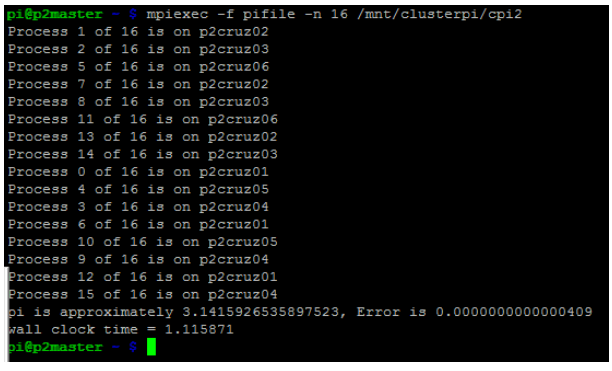

Figura 6. *Ejecución del programa cpi2 en el clúster "CruzII"*

Luego, ejecutamos el mismo programa en el Clúster Cruz II utilizando 6 nodos con 24 núcleos ver Figura 7:

```
pi@p2master ~ $ mpiexec -f pifile -n 24
/mnt/clusterpi/cpi2
```

Demoró en ejecutar un tiempo de 0,764868 segundos.

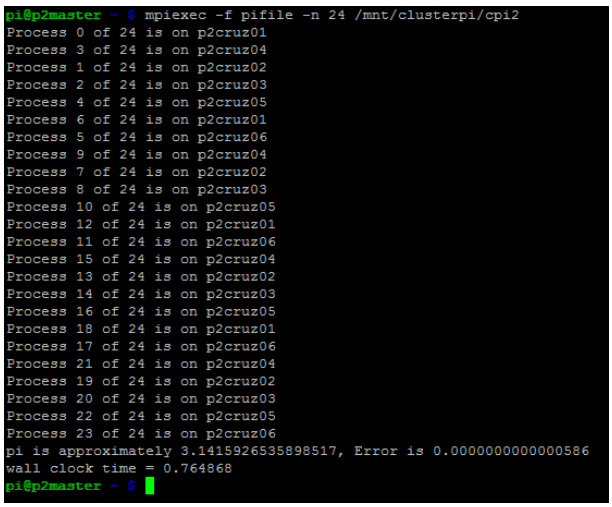

Figura 7. *Ejecución del programa cpi2 usando 24 núcleos en el clúster "Cruz II"*

5.2 Otro ejemplo de programa MPI

Ejecución del programa llamado "icpi" ver Figura 8:

```
ejecuta_mpi 16 /mnt/clusterpi/icpi
```

Con un número de intervalos de 1800 millones da como tiempo 16,877792 segundos. Y otro número de intervalos de 2000 millones da como tiempo 17,938480 segundos:

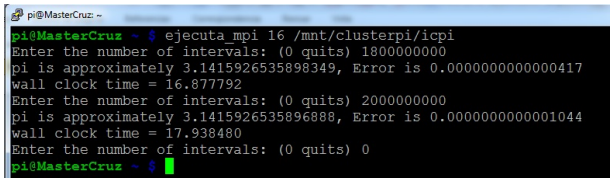

Figura 8. *Ejecución del programa icpi en el clúster "Cruz I"*

Ahora la ejecución del programa "icpi" en el clúster Cruz II ver Figura 9. Con 24 núcleos y 6 nodos:

```
pi@p2master ~ $ mpiexec -f pifile -n 24
/mnt/clusterpi/icpi
```

Con un número de intervalos de 1800 millones da como tiempo 7,033739 segundos. Y otro número de intervalos de 2000 millones da como tiempo 7,807122 segundos:

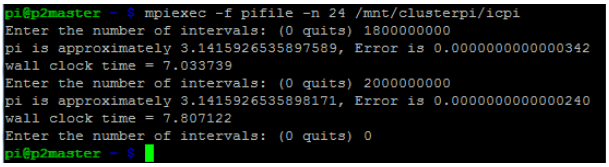

Figura 9. *Ejecución del programa icpi en el clúster "Cruz II"*

6 Comparación entre una PC y el clúster Cruz II

6.1 Ejecución del programa cpi2

La comparación del tiempo de procesamiento se realizó con una PC Intel Core i5 M 480 @ 2.67GHz. El programa "cpi2" que anteriormente habíamos ejecutado en los clústeres Cruz I y Cruz II, se ejecutó en una máquina virtual corriendo Ubuntu considerando hasta 2 núcleos físicos. Dando el siguiente resultado ver Figura 10: Se puede notar que se demoró en ejecutar un tiempo de 0,758889 segundos.

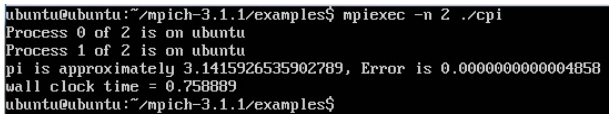

Figura 10. *Ejecución del programa cpi2 en una PC*

El resultado nos muestra que la PC para este programa en particular(cpi2) es ligeramente más rápida que el clúster Cruz II.

6.2 Ejecución del programa icpi

Igualmente ejecutamos el programa icpi en la PC, el que también habiamos ejecutado anteriormente en los clústeres Cruz I y Cruz II. Dando como resultado ver Figura 11.

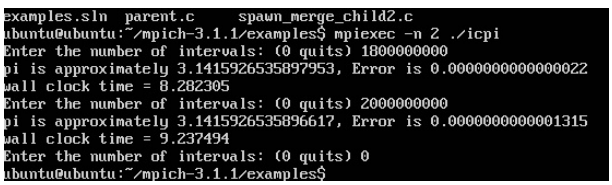

Figura 11. *Ejecución del programa icpi en una PC*

Notamos que la PC para este programa en particular(icpi) es ligeramente mas lento que el clúster Cruz II.

7 Conclusiones

Los resultados de las mediciones realizadas nos han permitido comprobar el aumento de potencia de computo considerable del Raspberry Pi versión 2 frente al Raspberry Pi B+ y de esa manera podemos contar con una información valiosa que nos permita de esa manera situarnos en el contexto real cuando usamos el Raspberry Pi. A pesar de la potencia de computo todavía limitada se considera a este equipo de mucha utilidad en el campo educativo. Como se comprobó anteriormente con el uso del clúster Cruz I, este nuevo clúster nos sirve igualmente para la enseñanza y el aprendizaje de la programación paralela. Además, nos permite probar e instalar otros tipos de software para el uso de la tecnología Big Data y otros.

8 Agradecimientos

Agradezco a Dios, la UNI, CTIC y a la Facultad de Ciencias.

9 Apéndices

9.1 Programa cpi2.c

```
#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 = 3.14159265358979323846264 3;
double mypi, pi, h, sum, x;
double startwtime = 0.0;
double 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 = 180000000;  /* # de rectángulos aumentado */
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) {
  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;
}
```

9.2 Programa icpi.c

```
/* Esta es una versión interactiva del \
programa 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);
 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;
}
```


1. Cesar Martin Cruz Salazar. *CRUZ: un Cluster aplicado a la educacion, portatil, de bajo costo, usando Raspberry Pi*. REVCIUNI, Volumen 16, páginas 1-6, 2013.

2. Simon J. Cox. *Steps to make a Raspberry Pi Supercomputer*. http://www.southampton.ac.uk/∼sjc/raspberrypi/, 2014.

3. Tina Chen. *Instalación del NFS server*. http://experimentandoconraspberrypi.blogspot.com/2013/06/compartir-directorio-con-nfs.html, 2013.

4. Andrew K. Dennis. *Raspberry Pi Super Cluster*. 2013.

5. Nikilesh Balakrishnan. *Building and benchmarking a low power ARM cluster*. 2012.

6. Jimbo Wales. *Benchmark*. https://es.wikipedia.org/wiki/Benchmark_(inform%C3%A1tica)

7. J. Dongarra A. Cleary A. Petitet, R. C. Whaley. *HPL - A Portable Implementation of the High Performance Linpack Benchmark for Distributed Memory Computers*. https://www.netlib.org/benchmark/hpl/

8. Akshay Pai. *HPL (High Performance Linpack): Benchmarking Raspberry Pis*. https://www.howtoforge.com/tutorial/hpl-high-performance-linpack-benchmark-raspberry-pi/

9. Advanced Clustering Technologies. *How do I tune my HPL.dat file?*. http://www.advancedclustering.com/act-kb/tune-hpl-dat-file/

10. Brian Dye *Distributed Computing with the Raspberry Pi*, 2014.