"CRUZ": un Clúster aplicado a la educación, portátil, de bajo costo, usando Raspberry Pi

César Martín Cruz Salazar
Escuela de Ciencia de la Computación. Facultad de Ciencias.
Universidad Nacional de Ingeniería;
ccruz@uni.edu.pe

Recibido el 21 de Junio del 2012; aceptado el 21 de Julio del 2012

Este artículo es sobre la implementación de un Clúster llamado CRUZ (cuyas siglas significan Clúster de Investigación de la Uni de potencia Zero) como herramienta educativa portátil formada de 16 nodos y un nodo adicional para el nodo maestro. Utiliza en total 17 placas Raspberry pi que se unieron formando una red cada uno con acceso a Internet. El equipo en su conjunto se configuró instalando software apropiado, de tal forma que trabajen en colaboración procesando una tarea particionada entre todos los nodos utilizando librerías MPI. Permitiendo así desarrollar y probar programas en paralelo. El objetivo de este equipo es que se, utilice como una herramienta para la enseñanza en los cursos de Programación Paralela y Distribuida, lo que proporciona un punto de partida de bajo costo para inspirar y capacitar a los estudiantes a entender y aplicar la computación de alto rendimiento y manipulación de datos para hacer frente a la ingeniería compleja y los desafíos científicos.

Palabras Claves: clúster, raspberry pi, programación paralela, programación distribuida, MPI, hadoop, clúster educativo, portátil, bajo costo.

This paper is about implementing a cluster called CRUZ (Cluster Research of the Uni of Zero power) as an educational tool portable consists of 16 nodes and an additional node for the master node. It uses a total of 17 Raspberry Pi boards joined to form a network with access to the internet. The equipment as a whole was configured by installing appropriate software, working in collaboration processing a partitioned task between all nodes using MPI libraries, thus allowing developing and testing parallel programs. The purpose of this equipment is to be used as a tool for teaching courses in Parallel and Distributed Programming, which provides a starting point inexpensive to inspire and enable students to understand and apply high performance computing and data handling to tackle the complex engineering and scientific challenges.

Keywords: cluster, raspberry pi, parallel computing, distributed computing, MPI, hadoop, education cluster, portable, low cost.

1. Introducción

La motivación inicial fue impulsar el uso de los Raspberry Pi (RPI) en la escuela de Ciencia de la Computación de la Facultad de Ciencias de la UNI y promover la enseñanza y aprendizaje de la computación de alto rendimiento (HPC) usando estas placas interconectados como Clúster. Entonces, empezamos investigando en internet experiencias relacionadas con Clústeres basados en RPIs, encontramos además de las desarrolladas en Reino Unido y Estados Unidos unos videos en youtube [1] y [2] que nos dieron algunas luces de cómo construir un Clúster. La información encontrada en Internet y libros si bien es cierto nos ayudó a construir el Clúster, no fue suficiente para darle al Clúster una aplicación de una verdadera herramienta para la enseñanza y el aprendizaje que es lo que perseguíamos, porque para que el Clúster tenga esta utilidad hemos tenido que instalar y desarrollar software adicional que no se menciona en la información revisada y también realizar configuraciones adicionales. Estas tareas adicionales las consideramos nuestro aporte y sobre las mismas tratará este artículo. Es así que en este artículo presentamos los desarrollos adicionales realizados en el Clúster "Cruz"que lo convirtieron en una herramienta para la enseñanza y aprendizaje de la programación paralela. El Clúster fue montado usando 17 placas Raspberry Pi interconectadas con enlaces Ethernet de 100 Mbits/s. Dos racks que comprende cada uno 7 placas RPIs y van conectados a switches se muestra en la Figura 1. El peso ligero y pequeño volumen, la hacen eminentemente adecuado para una serie de aplicaciones que demandan la programación paralela y que ejecuta un Clúster convencional pero que debido a sus altos costos y requisitos especiales de infraestructura su uso para la enseñanza resulta poco adecuado. Fue una gran fuente de inspiración el desarrollo de una supercomputadora basado en 64 RPIs desarrollado por un equipo liderado por el Prof. S. J. Cox en la Universidad de Southampton del Reino Unido [3].

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 un área de gran interés, tanto en el ámbito académico como en la industria. Aplicaciones ricas en datos son cada vez más frecuentes, los proveedores de infraestructura están prestando considerable

atención tanto a la energía consumida por el hardware computacional como la capacidad de enfriamiento que esto impone. La industria del hardware está ofreciendo más servidores de bajo consumo energético, a menudo usando CPUs de bajo consumo, y existe un creciente interés en el uso de chips de bajo consumo de arquitecturas alternativas tales como ARM en el centro de datos; empresas como Calxeda se han fundado en torno a estos desarrollos. Por otra parte, "Parallella: Un supercomputador para Todos"del proyecto Adapteva [4], recientemente logró reunir suficiente financiamiento comunitario que le permitirá lograr el objetivo de democratizar la computación de alto rendimiento mediante la producción de placas compactas basadas en un conjunto escalable de procesadores RISC sencillas acompañadas de una CPU ARM. En el centro de supercomputadoras de San Diego [5] usan el clúster "METEOR"basado en 16 nodos RPIs para la enseñanza y difusión de la programación paralela.

Figura 1. *Dos pilas(racks) del Clúster Cruz conectados a switches*

2. El presente trabajo

El Clúster mostrado en la Figura 2 que presentamos en este trabajo combina los elementos no convencionales del 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 trabajo realizado por Joshua Kiepert [6] donde se muestra un clúster de 32 nodos más un nodo maestro usado en sus pruebas para la disertación de su tesis de doctorado. 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. Con un tamaño global muy compacto, de peso ligero y de una gran portabilidad nuestro clúster es ideal para propósitos educativos y de demostración. Se dan en la Sección 3 las especificaciones del Clúster, describiendo su hardware y software. En la Sección 4 dirigimos nuestra atención a los aportes que hemos desarrollado y a la ejecución de ejemplos de programas en paralelo en el Clúster.

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"se compone de 16 Raspberry Pi entre Modelo B y B+ 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, lo cual permite su fácil manipulación y son una clase de computadores de una sola placa. Cada uno cuenta con 512 Mbytes de RAM, un sistema sobre un Chip(SoC) Broadcom BCM2835, que integra un procesador de 700MHz ARM1176JZF RISC, 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 dos puertos USB 2.0. El almacenamiento local se realiza a través de una tarjeta de memoria SD (Secure Digital) o microSD según el modelo de 8 gigabytes que se coloca en una ranura acondicionada para la tarjeta de memoria, y hay varios otras interfaces como buses I2C y SPI, UART y ocho pines GPIO. Cada placa RPI se alimenta con una fuente de 5 voltios que son suministrados por dos tipos de fuentes una de ellas es una fuente switching de computadora PC y los otros son adaptadores de voltaje cada uno con dos sockets USB. Comprende de 2 ventiladores para refrigeración que son necesarios porque cada placa RPI se encuentra overclockeado a 1 Ghz y 1 disco duro de 160Gbytes para almacenamiento de programas y datos.

Figura 2. *El Clúster "Cruz"*

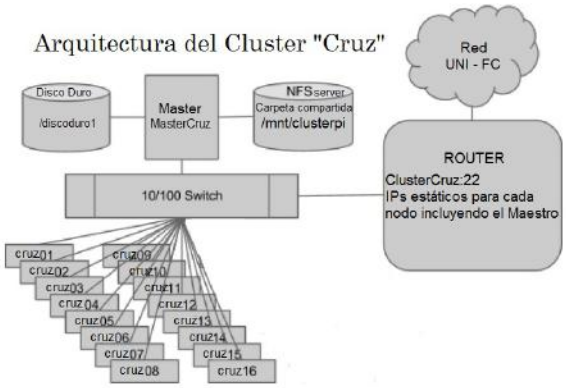

Figura 3. *Arquitectura del Clúster "Cruz"*

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 SD(MicroSD) con una imagen descargada de la Fundación Raspberry Pi(raspberrypi.org), y personalizamos esta instalación añadiendo software necesario para ejecutar programas en paralelo. Esta finalmente viene a ser la imagen del "maestro". Luego, hemos clonado esta imagen "maestro", a las tarjetas SD de los demás nodos. Para configurar el Clúster se siguieron los pasos mencionados en la referencia [7] y los pasos indicados en la referencia [8].

3.2.1. Instalación y configuración del NFS server

La documentación que encontramos sobre experiencias de desarrollo de Clústeres con RPIs no menciona esta instalación. Algunas referencias indican que se debe copiar el mismo archivo ejecutable en todos los nodos. Investigando sobre los Clústeres clásicos vimos que utilizaban el NFS server para compartir archivos. Es así que consideramos necesario y útil realizar la instalación del NFS server y su posterior configuración, siguiendo la referencia [9]. De este modo se pudo compartir la carpeta /mnt/clusterpi del nodo Maestro entre todos los demás nodos. Esta carpeta se utiliza para guardar los programas que se van a ejecutar en todos los nodos del Clúster. Para observar el estado del NFS ingresamos en línea de comando:

```
/etc/init.d/nfs-kernel-server status
```

Para habilitar desde el arranque el inicio del NFS ingresamos:

```
sudo update-rc.d rpcbind enable
```

3.2.2. Creación de un usuario sin derecho de administración

Como el objetivo es que sea utilizado también por los estudiantes es que creamos un usuario `mcruz` con contraseña `mcruz` para que cualquier estudiante pueda ejecutar programas de ejecución paralela en el clúster, pero sin correr el riesgo de que pueda borrar archivos del sistema e instalar nuevo software en el Clúster. Para lograr ello, tuvimos que crear el usuario mcruz en cada nodo y ejecutar para todos los nodos lo siguiente, por ejemplo para el nodo cruz12:

```
ssh-keygen -t rsa -C ce.cruz@gmail.com
```

Luego, en el nodo maestro:

```
cat ~/.ssh/id_rsa.pub | ssh \
cruz12 " cat >> /home/mcruz/.ssh/authorized_keys"
```

Estos pasos anteriores se hacen con el propósito de que cuando se ejecute el programa en cada nodo no solicite el password cada vez.

3.2.3. Cambiar los IPs a IPs estáticos en todos los nodos

Para lograrlo se cambió en cada nodo del Clúster el contenido del archivo /etc/network/interfaces.

3.2.4. Programa de chequeo de conexión hecho en Python y utilizando Crontab

Un problema muy común que apareció en el Clúster era la pérdida de conexión de cualquiera de sus nodos y este hecho se debe a que el nodo por alguna razón se cae. Entonces, nos planteamos desarrollar un programa en Crontab que chequee cada 20 segundos la conexión en Red de cada nodo y que en caso se pierda la conexión se reconecte en forma automática sin que alguien tenga que venir a reiniciarlo en forma manual. El programa se hizo en Python (checkInternet) y chequea la conexión con Google. Si se pierde, el programa manda a reiniciar el nodo en forma automática.

3.2.5. Un gestor de colas

Para evitar ejecutar en el Clúster más de un programa de manera simultánea, es que escribimos un script que chequea si el Clúster ya está corriendo un proceso. Si fuera así el proceso siguiente que quiere ejecutarse se pone en estado de espera hasta que finalice el anterior proceso.

```
ejecuta_mpi 4 /mnt/clusterpi/cpi2
```

Ver Figura 4 para una demostración de la ejecución de este script.

Este script no chequea que nodos están disponibles para lanzar otro programa que quiere ejecutarse. Una futura mejora al script debería chequear los nodos disponibles para que si los haya se lance otro programa que requiera estos nodos.

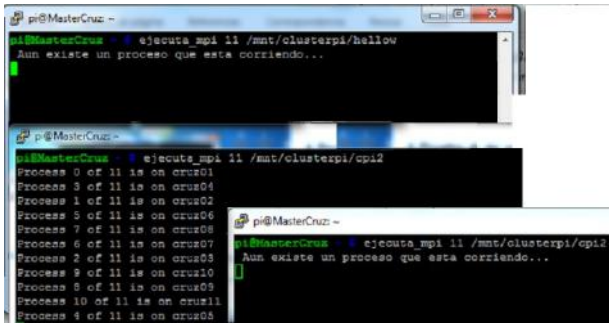

Figura 4. *Ejecutando a la vez en tres terminales el script*

3.2.6. Programas desarrollados en python para el clúster

En un clúster es necesario saber si todos los nodos se encuentran en red, antes de lanzar un programa a ejecutar. Para ello se desarrolló el programa `conectados.py` que chequea si los nodos se encuentran conectados en Red.

```
#! /usr/bin/python
#PROGRAMA que detecta la conexión de la red en \
el Cluster "Cruz".
import socket
import os
def get_cpu_temp():
tempFile = \
open( "/sys/class/thermal/thermal_zone0/temp" )
cpu_temp = tempFile.read()
tempFile.close()
return float(cpu_temp)/1000
print '                                    '
print 'CLUSTER "CRUZ". 16 Nodos conectados + \
 Maestro',"(Temperatura=",get_cpu_t$, ")"
print '====================================\
==========='
print "                  "
print "      IP      ", "      HOST", "   Estado RED"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(0.5)
try:
s.connect(('172.20.45.246',22))
print '172.20.45.246  ----  cruz01  OK en RED'
except Exception, e:
print '172.20.45.246  ----  cruz01  FALLO en la RED'

s.close()
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(0.5)

try:
s.connect(('172.20.45.247',22))
print '172.20.45.247  ----  cruz02  OK en RED'
except Exception, e:
print '172.20.45.247  ----  cruz02  FALLO en la RED'

s.close()
.....................................................
.....................................................
#Este bloque de código se repite para cada nodo.
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(0.5)
try:
s.connect(('172.20.45.230',22))
print '172.20.45.230  ----  cruz16  OK en RED'
except Exception, e:
print '172.20.45.230  ----  cruz16  FALLO en la RED'

s.close()
```

Ver la ejecución de este programa en la Figura 5. Igualmente, se necesita conocer la temperatura de los nodos del clúster, entonces se desarrolló el programa `temperaturas.py` :

```
#PROGRAMA que visualiza la temperatura del Cluster \
 "Cruz".
#Hecho por: Martin Cruz Salazar
import os
print "                          "
print "TEMPERATURAS DEL CLUSTER CRUZ"
print "============================="
print "                            "
os.system("python tempMaestro.py")
os.system("ssh cruz01 python temperatura.py exit")
os.system("ssh cruz02 python temperatura.py exit")
    ---------------------------------------------
    ---------------------------------------------
os.system("ssh cruz14 python temperatura.py exit")
os.system("ssh cruz15 python temperatura.py exit")
os.system("ssh cruz16 python temperatura.py exit")
```

El programa `temperatura.py` se encuentra en cada nodo, como ejemplo en el nodo1:

```
#Programa que visualiza la temperatura de
#este nodo. Hecho por: C. Martin Cruz
import os

def get_cpu_temp():
tempFile = open( "/sys/class/thermal/thermal_zone0/temp" )
cpu_temp = tempFile.read()
tempFile.close()
return float(cpu_temp)/1000
# Return % of CPU used by user as a character string
def getCPUuse():
return(str(os.popen("top -n1 | awk '/Cpu\(s\):/ {print $2}'")
)))
print "Nodo cruz01 ==>"," Temperatura=",get_cpu_temp()
```

Ver la ejecución del programa en la Figura 6.

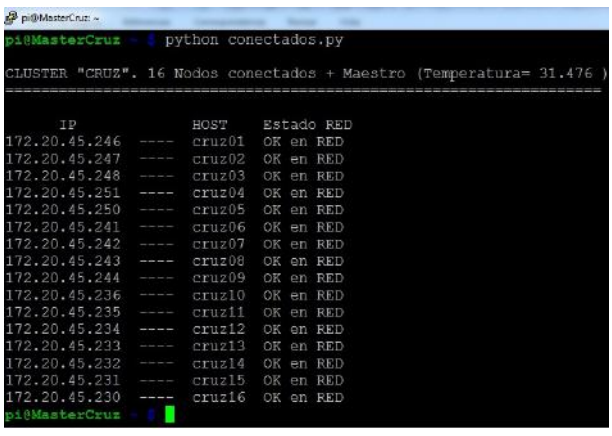

Figura 5. *Ejecución del programa conectados.py en el clúster "Cruz"*

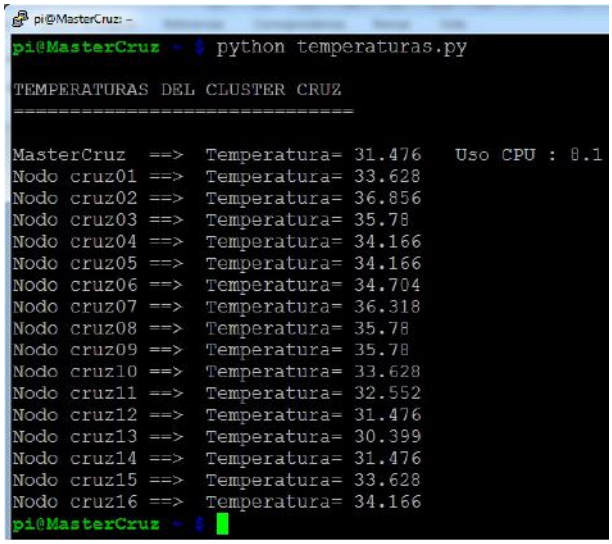

Figura 6. *Ejecución del programa temperaturas.py en el clúster "Cruz"*

4. Probamos programas en el Clúster

Los programas que se ejecutarán vienen como ejemplos al descargar el archivo de instalación de MPI:

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

4.1. Ejecución del programa cpi2

En la Figura 7 se muestra como se ejecuta el programa "cpi2" de cálculo del valor de "pi" que ha sido modificado para considerar un número de intervalos de 180millones y el resultado obtenido considerando 16 nodos. El archivo se encuentra en la carpeta compartida /mnt/clusterpi. Para ejecutar el programa en la línea de comando ingresamos:

```
mpiexec -f pifile -n 16 /mnt/clusterpi/cpi2
```

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

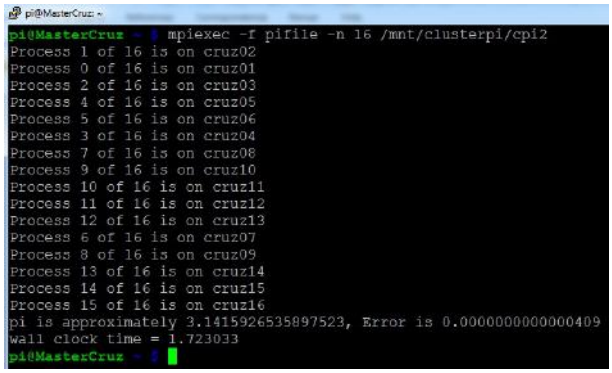

Figura 7. *Ejecución de un programa cpi2 con MPI en el clúster "Cruz"*

Ejemplo de ejecución del mismo programa usando el script Gestor de colas y 12 nodos ver Figura 8:

```
ejecuta_mpi 12 /mnt/clusterpi/cpi2
```

Demoró en ejecutar un tiempo de 2,221911 segundos.

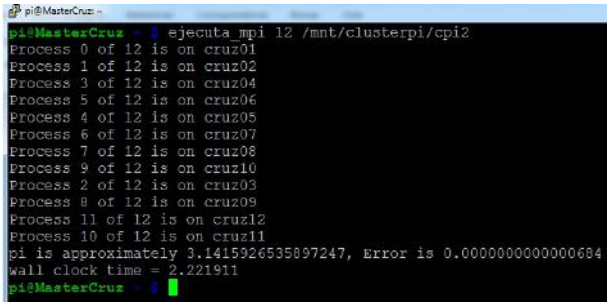

Figura 8. *Ejecución de un programa con MPI usando el script gestor de colas en el clúster "Cruz"*

4.2. Otro ejemplo de programa MPI

El programa "icpi" ver Figura 9:

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

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

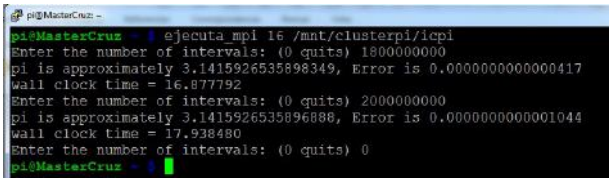

Figura 9. *Ejecución del programa icpi con MPI en el clúster "Cruz"*

5. Comparación entre una PC y el Clúster

5.1. Ejecución del programa cpi2

La comparación del tiempo de procesamiento se realizó con una PC Intel Core i5 M 480 @ 2.67GHz. Se ejecutó el programa cpi2 que se ejecutó anteriormente en el Clúster, 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.

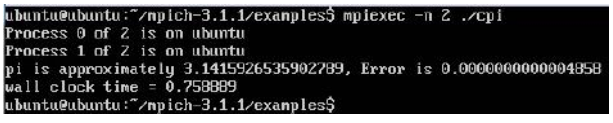

Figura 10. *Ejecución del programa cpi2 en una PC*

Resulta que la PC para este programa en particular es 2,29 veces más rápida que el Clúster.

5.2. Ejecución del programa icpi

Se ejecutó el programa icpi en la PC, el que se ejecutó anteriormente en el Clúster. Dando como resultado ver Figura 11.

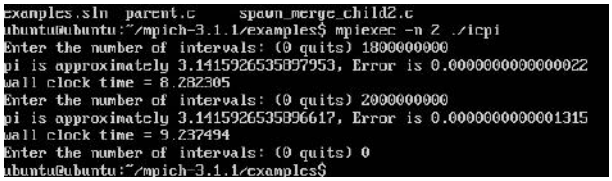

Figura 11. *Ejecución del programa icpi en una PC*

Se puede ver que la PC para este programa en particular es alrededor de 2 veces más rápida que el Clúster.

6. Conclusiones

Es un equipo de mucha utilidad a pesar que no tiene una gran potencia de procesamiento como se comprobó al compararlo con la potencia de procesamiento de una PC con procesador Intel i5 con 2 núcleos físicos. Pero la utilidad está en que además de permitirnos ejecutar programas en paralelo, se puede observar la latencia y servir para la enseñanza y el aprendizaje. Además, nos ha permitido poder desarrollar otros programas en Python que son necesarios para un Clúster de este tipo. Se ha pensado en un futuro enriquecer el clúster con un software que permita ver el estado actual de funcionamiento del Clúster en la web, así como la instalación y configuración de un gestor de colas avanzado que será motivo de otro artículo.

Agradecimientos

Agradecemos primero a Dios por permitirnos ejecutar este proyecto, a la UNI y a la Facultad de Ciencias por el espacio otorgado. A los profesores Fidel Jara, Carmen Eyzaguirre por el aliento dado para terminar el proyecto. Al profesor Renato Tovar por sus consejos y sus conocimientos sobre Clústeres. Al profesor Glen Rodríguez porque se animó a probar el Clúster en su curso de programación paralela, haciendo así el control de calidad del Clúster. A los profesores Hector Loro y Oswaldo Velásquez por la corrección y revisión de este artículo. A los estudiantes Abraham Zamudio, Pavel Mendoza Villafane, Jean Pierre Huaroto Chavez y Diego Berrocal Chinchay porque en alguna etapa del desarrollo del Clúster tuvieron cada uno un aporte valioso. Finalmente a los alumnos del curso CC571 ciclo 2014 II, por la ejecución del proyecto de Crontab.

1. How to make a cluster computer (part 1). `http://www.youtube.com/watch?v=1R0UgIgcb5g`, 2014.
2. How to make a cluster computer (part 2). `https://www.youtube.com/watch?v=lHmFRlETTcQ`, 2014.
3. Simon J. Cox, James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott, and Neil S. O'Brien. Iridis-pi: a low-cost, compact demonstration cluster. *Cluster Computing*, 17(2):349–358, 4 2014.
4. Parallela un supercomputador para todo el mundo. `https://www.kickstarter.com/projects/adapteva/parallela-a-supercomputer-for-everyone`, 2014.
5. Sdsc uses meteor raspberry pi cluster to teach parallel computing. `http://ucsdnews.ucsd.edu/pressrelease/sdsc_uses_meteor_raspberry_pi_cluster_to_teach_parallel_computing`, 2014.
6. Rpicluster de la universidad del estado de boise. `http://coen.boisestate.edu/ece/raspberry-pi/`, 2014.
7. Steps to make a raspberry pi supercomputer. `http://www.southampton.ac.uk/~sjc/raspberrypi/`, 2014.
8. Andrew K. Dennis. *Raspberry Pi Super Cluster*. 2013.
9. Instalacion del nfs server. `http://experimentandoconraspberrypi.blogspot.com/2013/06/compartir-directorio-con-nfs.html`, 2013.