March 9, 2016
[FRANCISCO JAVIER MARTÍN COTEÑO
CARLOS ESTEBAN SERRANO CORTÉS
2 ASIR

Proyecto Servidor Cluster

CPTN SYSTEMS

Índice

1. Introducción

• Objetivo .....(Página **2**)

• Justificación .....(Página **3**)

• Análisis de lo existente .....(Página **4**)

• Materiales necesarios ..... (Página **5**)

2. Configuración del cluster

• Instalación del sistema Operativo (Ubuntu Mate).....(Página **7**)

• Configuración básica del cluster.....(Página **10**)

3. Configuración de servicios en cluster

• Servicio FTP mirror.....(Página **15**)

• Servicio Proxy Balancer sobre Apache.....(Página **18**)

• Shell Script para Administración del cluster.....(Página **20**)

4. Servidor ecológico

• Montaje y configuración de la parte eléctrica.....(Página **24**)

5. Posibles mejoras

• Mejoras de parte eléctrica.....(Página **29**)

• Mejoras parte informática.....(Página **30**)

1. Introducción

Objetivo

Este proyecto nació con la idea de poder crear un servidor eficiente, económico, totalmente funcional y a la vez respetuoso con el medio ambiente.

Queremos demostrar que se pueden ofrecer unos servicios competentes con una inversión inicial reducida y un coste energético mínimo.

Aunque a un nivel funcional menor ya que nuestro capital es limitado, intentaremos verificar que con un gasto reducido se puede llegar a crear un servidor autosuficiente y perfectamente operativo.

Esto se conseguirá gracias a que aprovecharemos la luz solar con unas placas que lograrán abastecer de energía limpia nuestro servidor y sus periféricos o al menos una parte de ellos en nuestro modelo.

Nuestro proyecto está orientado a las pymes y también a las grandes empresas, ya que ambas buscas un producto de calidad, de reducidas dimensiones y que tenga coste económico mucho más bajo que los servidores convencionales actuales. Gracias al sistema de cluster nuestro servidor no va a carecer de potencia respecto a otras máquinas existentes en el mercado.

Nuestro objetivo principal es probar que con poco dinero y usando la tecnología Rasberry, podemos dar servicio a las empresas, cubriendo así todas sus necesidades informáticas (alojamiento web, bases de datos ftp..) y de una manera económica y funcional.

Finalmente el proyecto acabado será un servidor cluster de 2 Raspberrys PI2, con 3 servicios en cluster (FTP, Proxy Balancer y BBDD MYSQL). Todo ello presentado dentro de una caja de conexiones con su sistema eléctrico y que se alimentará gracias a la energía solar con una placa y un acmulador.

Justificación

La principal razón de la realización de este proyecto fue la búsqueda de una alternativa económica a los precios de los actuales servidores. Queremos demostrar que con nuestras Raspberrys Pi 2 somos capaces de montar y configurar un servidor que dará prestaciones similares a la de los servidores convencionales.

Todo esto lo haremos manteniendo nuestras premisas del ahorro de costes en el hardware y mantenimiento.

Sin querer contentarnos solo con esto, hemos querido añadir un módulo adicional a nuestro proyecto, que será capaz de proporcionarnos la energía que necesita para trabajar, gracias a las placas solares que implementaremos en nuestro proyecto.

Por tanto la justificación principal es el de llenar un hueco en el mercado tecnológico para que las empresas puedan tener servidores eficientes, potentes y a un bajo coste.

Análisis de lo existente

Desde el primer momento antes de plantearnos meternos a fondo con la realización del proyecto, estuvimos investigando por internet si existía un producto igual que el nuestro, las conclusiones fueron negativas. Encontramos varios proyectos de tipo educativo como el de la Universidad de Southampt(www.shothampton.ac.uk/rasberrypi/) que principalmente se centra en demostrar el potencial de la Rasberry Pi en modo cluster pero a nivel educativo y con un software propietario para gestionar los nodos de las Rasberrys.

También encontramos un prototipo de servidor cluster creado por la filial de Google "Kubernetes", que hizo una demostración con varias Rasberrys Pi 2 testeando su software para clusterizar servicios, escribieron un blog con un par de entradas detallando a groso modo su proyecto (blog.kubernetes.io). Pero tampoco nos pareció que fuese algo que se pareciese ni remotamente a nuestro proyecto, ya que esto es más una demostración, un experimento para verificar que su software funciona también en Rasberry.

Por último pudimos ver como la compañía NVIDIA que se dedica principalmente al diseño y fabricación de tarjetas gráficas, en el año 2013 realizó un super computador con varias Rasberrys en serie. Pero este proyecto no fue con fines empresariales, sino como un artículo que colgaron en su blog (blogs.nvidia.com). A esto le añadiremos que para controlar sus servicios no lo hicieron ellos "manualmente" (entiéndase por manualmente el configurar uno a uno los servicios a clusterizar desde su archivo .conf) sino que compraron un software llamado HPC que controlaba y clusterizaba automáticamente los servicios.

Nosotros en cambio, ofrecemos un producto 100% configurado por nosotros, sin otros software de pago que administren los servicios del cluster y todo ello bajo el sistema operativo gratuito Ubuntu Mate.

Propuesta detallada

Vamos a configurar un servidor en cluster con dos Rasberry Pi 2, en las máquinas vamos a instalar el sistema operativo Ubuntu Mate, implementaremos los siguientes servicios de manera redundante en ambas máquinas:

1 **Servidor Web** con balanceo de carga (cuando el servidor web reciba varias peticiones simultáneas de clientes, responderá una máquina y si está ocupara responderá la otra).

2 **Servidor FTP** con mirroring (cuando subamos archivos al ftp en una máquina, esa misma información será "reflejada" a modo de espejo en la otra máquina, conteniendo así los archivos en ambas Rasberrys).

Además, vamos a alimentar las Rasberrys con una placa solar de manera que crearemos un servidor que se coja la electricidad gracias a la energía del sol (evidentemente y por la limitación económica que tenemos, lo haremos a escala y demostraremos con una pequeña placa solar que eso es factible).

Elementos necesarios para hacer el cluster del servidor:

- 2 Raspberrys Pi 2
- 2 Tarjetas Micro SD 16 GB
- 2 Cargadores
- 1 Cable HDMI a DVI
- 2 cables rj45
- 1 Teclado y 1 ratón
- Sistema operativo Ubuntu Mate

Elementos necesarios para implementar la energía solar:

- Ventilador de ordenador
- Amperímetro para ver la carga solar de la placa
- Protoboard, resistencias y un potenciómetro (para controlar la velocidad del ventilador)
- Tornillos y tuercas
- Regulador de carga
- Trasnformadores de 12v a 5v con fusibles de 3amp
- Batería lipo de 500 mah
- Una placa solar de 380 ma
- Aislante
- Bridas
- Cinta aislante
- Transformador de corriente de 12 v

2. Configuración del cluster

Instalación del sistema operativo

Hemos elegido instalar como sistema operativo Ubuntu Mate, que es un S.O muy parecido a Ubuntu, pero adaptado especialmente para las Raspberrys. Por lo tanto descartamos elegir Raspbian ya que Ubuntu Mate ofrece muchas más características, es mucho más configurable y potente. Pero en contraposición Ubuntu Mate es más complejo de utilizar y existe mucho menos documentación que consultar que el Raspbian.

Bien ahora pasaremos a describir paso a paso la instalación básica del sistema operativo y la configuración de las tarjetas sd que harán de disco duro en las máquinas.

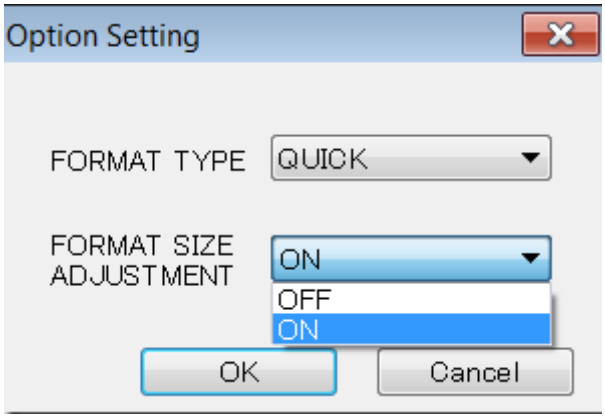
Configuración de las tarjetas SD:

Adquirimos 2 tarjetas de 32gb cada una para instalar el S.O y sus servicios y actualizaciones correspondientes. El primer paso a ejecutar es descargar la imagen del Ubuntu Mate desde el siguiente enlace *https://ubuntu-mate.org/raspberry-pi/*

Después tenemos que darle formato a las tarjetas sd y hacerlas un "resize" para que tengan disponible los 32gb y no tengamos problema después de meter la imagen. Por tanto descargamos el programa desde el siguiente enlace *https://www.sdcard.org/downloads/*

Este proceso es muy sencillo, insertamos la tarjeta sd en nuestro equipo, abrimos el SD Formater y seleccionamos "Opciones"

Ahora tenemos que elegir "Format type: Quick" y "Format type auto-adjustement: ON", esto hará un formateo rápido de la SD y lo hará con todo su tamaño disponible.



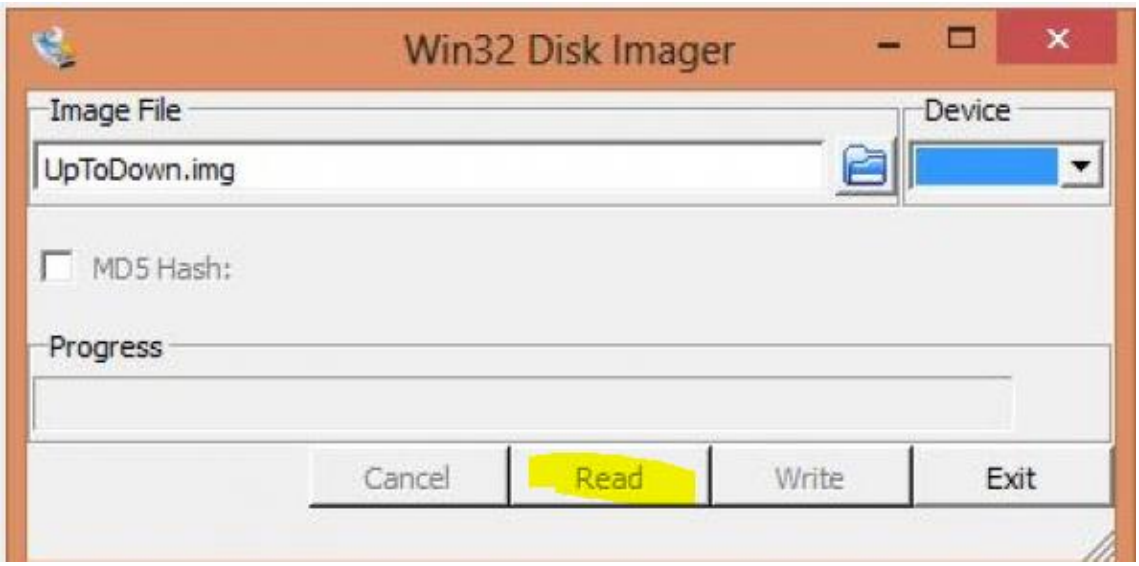
Repetiremos este mismo proceso en la otra tarjeta SD y una vez las tengamos listas procederemos a instalar Ubuntu Mate.

Como la imagen del S.O viene con una extensión .img, vamos a necesitar el programa "Win32DiskImager" para poner el Ubuntu en las memorias SD.

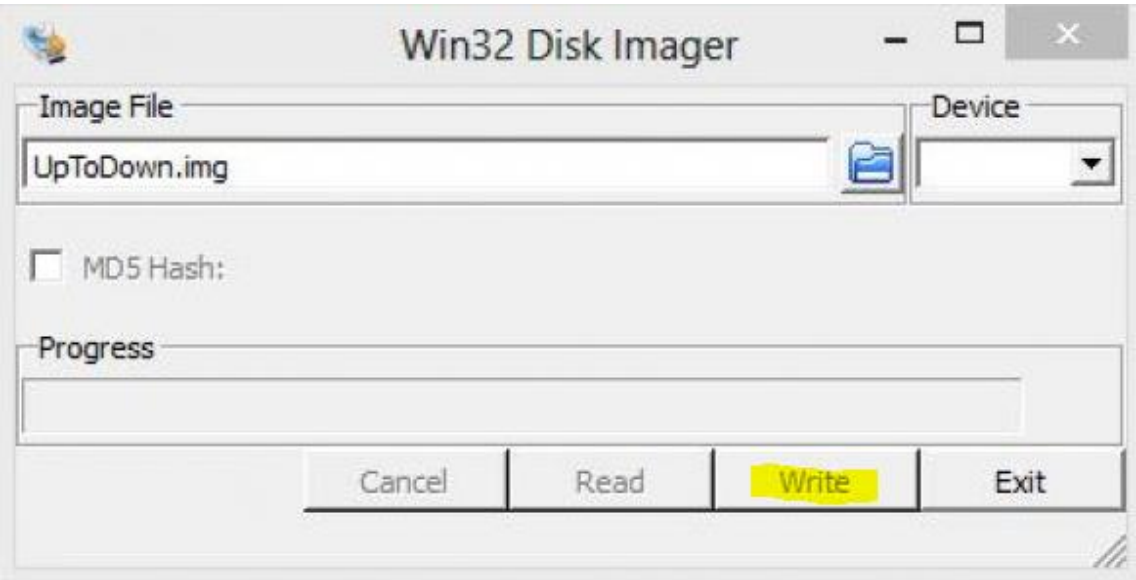
Por tanto, descargamos el programa del siguiente enlace:

http://win32-disk-imager.uptodown.com/

Una vez instalado, lo ejecutamos y seleccionaremos la imagen del Ubuntu, mediante la opción "Read"



Ahora seleccionaremos nuestra tarjeta SD, en el apartado "Device" y haremos click en la opción "Write".



Después de esto tendremos instalado en nuestras tarjetas el sistema operativo Ubuntu Mate y listas para empezar con la configuración de nuestro cluster.

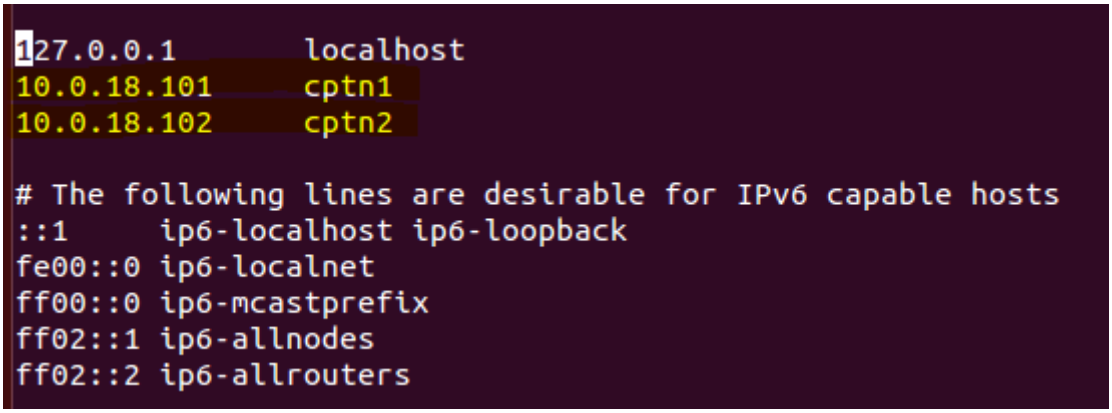
Configuración básica del cluster

Primeros pasos, configs básicas del sistema:

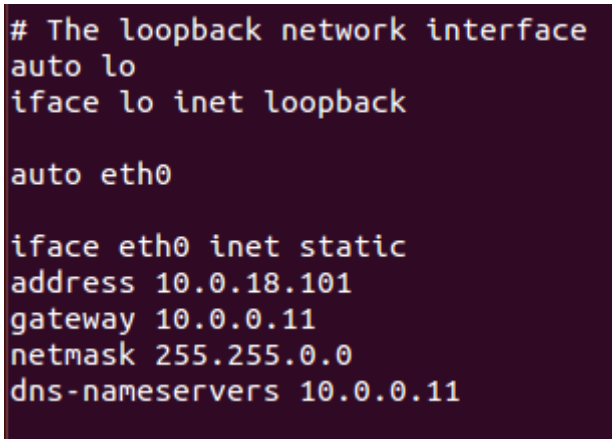
Antes de empezar a configurar el cluster para que las 2 máquinas trabajen como una sola, necesitamos hacer unas configuraciones básicas.

Vamos coger una sola Raspberry en ella haremos las configuraciones y después haremos una imágen para no tener que hacer el mismo trabajo en la otra máquina y ya simplemente cambiaremos sus parámetros como la ip, el nombre de máquina etc..

La primera Raspberry vamos a cambiarla el nombre, nosotros la hemos llamado "CPTN1", hacemos los cambios necesarios en el archivo /etc/hosts poniendo el nombre de máquina y la ip de ambas máquinas



Ahora tenemos que configurar la ip, como sale en la imagen vamos a darles las ip 10.0.18.101 y 10.0.18.102, para que estén en la misma red y se vean la una a la otra.



Una vez configurados estos aspectos básicos, haremos un update de la máquina para que tenga todas las actualizaciones disponibles instaladas y posteriormente reiniciaremos la máquina.

Configuración del cluster:

Gracias al update realizado anteriormente ya tendremos instalado el servicio ssh, por lo que podremos acceder a nuestra máquina de manera remota.

Ahora vamos a necesitar instalar un programa llamado "mpich", que es un Interfaz de Paso de Mensajes es un estándar que define la sintaxis y la semántica de las funciones contenidas en una biblioteca de paso de mensajes diseñada para ser usada en programas que exploten la existencia de múltiples procesadores.

El paso de mensajes es una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua, de manera similar a como se hace con los semáforos, monitores, etc.

Es decir con esto vamos a conseguir que se sincronicen los servicios y las dos máquinas trabajen como un cluster.

Crearemos un archivo llamado "mpich2" (*mkdir /mpich2)* para meter el mpich, hacemos un wget para descargar la última versión del mpich

```
wget http://www.mpich.org/static/downloads/3.1/mpich-3.1.tar.gz
```

Desempaquetamos el tar y ejecutamos el archivo de instalación, antes de esto necesitamos crear los siguientes directorios para que se instale correctamente:

```
sudo mkdir /home/rpimpi/

sudo mkdir /home/rpimpi/mpi-install

mkdir /home/pi/mpi-build

cd /home/pi/mpi-build
```

Y ahora por último ejecutamos el instalador

sudo /home/pi/mpich2/mpich-3.1/configure -prefix=/home/rpimpi/mpi-install

Por tema de ampliaciones futuras del servidor, vamos a instalar el "make" para poder programar tareas en phyton entre nuestro cluster

sudo make

sudo make install

Por último añadiremos al path el instalador del mpich, de la siguiente manera:

nano .bashrc

PATH=$PATH:/home/rpimpi/mpi-install/bin

Y haremos un reboot del sistema para que los cambios tengan efecto.

Vamos a instalar phyton para poder probar comandos con el mpich de manera que cuando tengamos las dos máquinas en cluster, podremos testear que ambas se ven y que responden como una sola, esto será un paso imprescindible para poder empezar a clusterirzar los servicios.

Descargamos y desempaquetamos el tar

sudo aptitude install python-dev

wget https://mpi4py.googlecode.com/files/mpi4py-1.3.1.tar.gz

tar -zxf mpi4py-1.3.1

Instalamos los archivos de phtyon que vamos a utilizar y lo metemos en el path para poder ejecutarlo

python setup.py build

python setup.py install

export PYTHONPATH=/home/pi/mpi4py-1.3.1

Ahora lo que vamos a hacer es crear una imagen de esta tarjeta con toda esta parte ya configurada y la pondremos en nuestra segunda Raspberry. Esto lo hemos hecho con el Win32 Disk-Imager que ya hemos descrito anteriormente.

Los únicos cambios que hay que hacer es cambiar a nuestra máquina el nombre a "cptn2", meter los datos en el archivo hosts (ya descrito anteriormente) y cambiar la ip a 10.0.18.102.

Configuración de las claves ssh para el cluster:

Primero vamos a añadir el programa nmap Sudo apt-get install nmap

Comprobamos que las ips son las correctas y vamos a crear un archivo llamado "Machine

Para que nuestras máquinas se "viesen" la una a la otra y poder correr procesos de manera simultánea en ambas máquinas, es necesario compartir una clave ssh en cada host y compartirlas.

Añadimos las keys a la "cptn1"

```
Ssh-keygen

Cd ~

Cd .ssh

Cp id_rsa.pub pi01

Ssh cptn1@10.0.18.102
```

Añadimos las keys a la "cptn2"

```
Ssh-keygen

Cd .ssh

Cp id_rsa.pub pi02

Scp 10.18.0.101:/home/pi/.ssh/pi01 .

Cat cptn1>> authorized_keys
```

```
Exit
```

Copiaremos las claves generadas mediante ssh de cptn1 a cptn2 y viceversa para que ambas máquinas las compartan y puedan trabajar entre sí, balancear servicios y responde como una sola máquina.

Ahora vamos a comprobar con la ejecución de un programa predeterminado del mpich, que ambas máquinas corren este programa a la vez y que se comunican la una con la otra.

Ejecutaremos desde la máquina cptn1:

```
scptn@cptn1:~$ mpiexec -f machinefile -n 2 hostname
cptn1
cptn2
```

Vemos que responden las dos máquinas y ejecutan el mismo programa que le hemos pasado por parámetro. Por lo tanto todo lo que hemos configurado está funcionando correctamente y ya tenemos terminada la configuración básica del cluster.

Ahora haremos una imagen de ambas Raspberrys para tener un backup de toda nuestra configuración y de este modo poder restaurar el sistema de una manera rápida y sin perder todo el trabajo anterior.

Para ello utilizaremos de nuevo el Win32 Disk Imager.

3. Configuración de servicios en cluster

Servicio FTP Mirror

Lo que queremos conseguir con este servicio es que el cluster tenga instalado un servicio de Transferencia de Archivos o FTP, configurado de tal manera que cuando en una máquina se cree un archivo lo haga a la vez en la otra, esto es lo que se llama Mirror (o espejo).

El primer paso es instalar un servicio ftp en ambas rasberrys (sudo apt-get install) proftpd, nosotros hemos elegido el "Proftpd" ya que es el que más posibilidades de configuración tiene.

Después tenemos que configurar el servicio abriendo su archivo de configuración (/etc/proftpd/proftpd.conf). Modificaremos varios parámetros:

Cambiar el nombre del servidor

```
ServerName                      "CPTN-SYSTEMS"
```

Comentar la parte de los usuarios anónimos para que no puedan hacer login en el servidor ftp

```
# <Anonymous ~ftp>
#   User                            ftp
#   Group                           nogroup
#   # We want clients to be able to login with "anonymous" as well as "ftp"
#   UserAlias                   anonymous ftp
#   # Cosmetic changes, all files belongs to ftp user
#   DirFakeUser on ftp
#   DirFakeGroup on ftp
#
#   RequireValidShell           off
#
#   # Limit the maximum number of anonymous logins
```

Configurar el mensaje de bienvenida al conectar al ftp y el mensaje de error al fallar esta conexión:

```
#Mensaje de bienvenida del servidor
AccessGrantMsg          "Bienvenido al ftp cluster de CPTN SYSTEMS"
#Mensaje de error de conexión
AccessDenyMsg           "Error de conexión al servidor"
```

No permitir que los usuarios tengan acceso a todo disco duro del servidor, sino solamente a su directorio home

```
# Use this to jail all users in their homes
DefaultRoot                     ~
```

Después copiaremos este archivo de configuración y mediante un scopy apuntando a la otra máquina lo copiaremos para tener la misma configuración en los dos servidores.

Ahora el reto es que lo que hagas en un servidor tenga consecuencias en el otro, es decir que si copias un archivo en cptn1 se copie en cptn2, es lo que llamamos "miroring". Para ello estuvimos investigando como poder hacerlo, primeramente nos fijamos en el servicio DFS de Windows que hace un miroring bidireccional, lo que haces en un servidor inmediatamente repercute en el otro, pero para nuestro sistema operativo (Ubuntu Mate) no existe algo parecido, hay que instalar y configurar un gestor de servicios para clusters y que gestione este servicio.

Investigando por la red, he descubierto un software desarrollado por Google para Linux, llamado "Kubernetes"(http://kubernetes.io/), que es un sistema que gestiona clusters y sus servicios, pero lamentablemente no está preparado para clusters de Raspberry Pi, ya que aun están en desarrollo.

Después seguimos investigando acerca de clusters de Raspberry y descubrimos que Nvidia había desarrollado uno, pero también utilizaba un software externo para gestionar los servicios, en este caso era con Ansible (www.ansible.com) pero su versión para Rasberry no es gratuita y había que desembolsar dinero para poder descargarla.

Así que nos decimos por utilizar un programa gratuito para Ubuntu, "lftp", que es un cliente ftp que soporta la shell Bash, cuya función es la de hacer un mirror de un host a otro. Por tanto implementamos un script para que llamara al ftp del otro servidor y realizase una copia exacta, pero solo lo conseguimos de manera unidireccional.

El "proftpd" fue instalado y configurado en cada máquina del cluster y creamos un usuario para poder operar con él "ftp-user". El script de mirroring quedó de la siguiente manera:

```
#!/bin/bash
#SCRIPT SINCRONIZACIÓN FTP CPTN SYSTEMS V 1.0
HOST='10.0.18.101'
USER='ftp-user'
PASS='ftpuser'
TARGETFOLDER='~'
SOURCEFOLDER='/home/scptn/ftpbackup'

lftp -f "
open $HOST
user $USER $PASS
lcd $SOURCEFOLDER
mirror --reverse --delete --verbose $SOURCEFOLDER $TARGETFOLDER
bye
"
```

Para automatizar esta tarea y que por tanto las copias en ambas máquinas se hagan de manera periódica, decidimos incluir este script en el crontab, ejecutándolo cada 30 minutos.

Servicio Proxy Balancer sobre Apache

Lo que vamos a hacer es configurar un Balanceo de carga entre el servidor Cptn1 y Cptn2, que dependiendo de la carga de peticiones o trabajo, cptn1 como master se encargará de gestionarlas y distribuirlas entre las 2 máquinas.

Teniendo la misma información en las webs el usuario final no se da cuenta de este balanceo, sin embargo para un servidor con mucha carga de trabajo esta ayuda puede ser fundamental para un correcto funcionamiento.

A pesar de no ser un cluster real se consigue ese efecto gracias al balanceo tipo proxy que nos concede el apache en su configuración.

Lo que hemos hecho es configurar en el archivo 000-default.conf 2 host virtuales uno con la ip 127.0.0.1 de la propia máquina "cptn1" para recibir sus propias peticiones y otra con el resto para poder balancear la carga entre ellas.

En este último caso hemos puesto las 2 ip fijas reales de las máquinas:

10.0.18.101 cptn1

10.0.18.102 cptn2

Lo que vamos a hacer es configurar un balanceador de carga mediante un proxy en nuestro servicio Apache2, con los mods que vienen ya instalados en el propio Apache, en la carpeta "mods-avialable" (etc/apache2/mods-avialable).

Lo primero que tenemos que hacer es activar estos mods con el comando "a2enmod", activaremos los siguientes mods:

a2enmod proxy, a2enmod proxy_http, a2enmod proxy_balancer, después reiniciaremos el servicio apache2.

El siguiente paso es configurar el balanceador para que reparta las peticiones de un servidor a otro, para ello tenemos que configurar adecuadamente el archivo 000-default.conf, vamos a poner las líneas de código más importantes:

```
<VirtualHost 127.0.0.1:80>
        DocumentRoot /var/www/html
</VirtualHost>
<VirtualHost *:80>
```

```
<Proxy balancer://cluster>
        BalancerMember http://127.0.0.1
        BalancerMember http://10.0.18.102
</Proxy>

 ProxyPass / balancer://cluster/ lbmethod=byrequests
```

Y ahora vamos a reiniciar el servicio y a ver su status para ver que está funcionando correctamente después de la configuración.

```
scptn@cptn1:/etc/apache2/mods-available$ sudo service apache2 restart
scptn@cptn1:/etc/apache2/mods-available$ sudo service apache2 status
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2)
   Active: active (running) since lun 2016-01-11 19:30:21 CET; 22s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 28588 ExecStart=/etc/init.d/apache2 start (code=exited, status=0/SUCCESS)
```

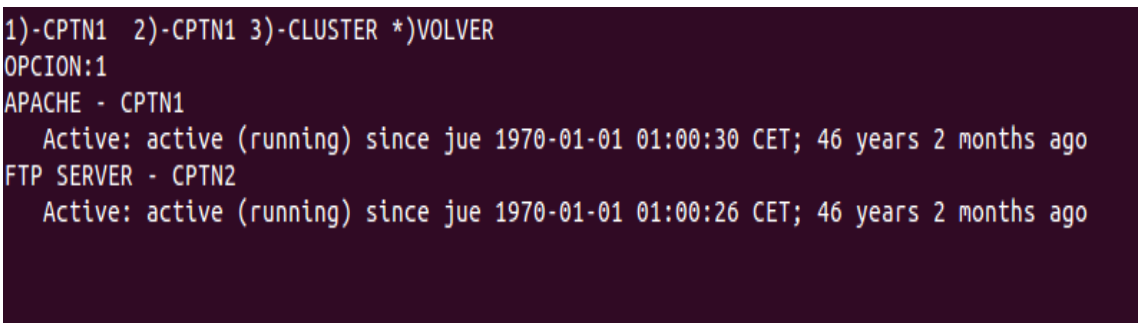
Después de esta configuración, cuando varios clientes hagan una petición a nuestra página web, responderá una máquina u otra dependiendo de su carga de trabajo.

Shell Script administración del cluster

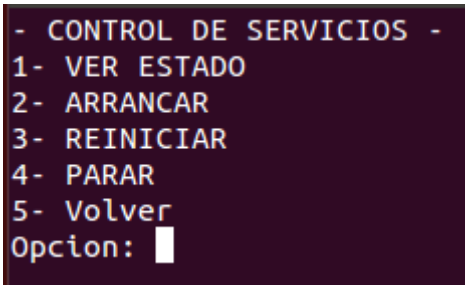
Para poder administrar todos los servicios de una manera centralizada decidimos hacer un script que nos permita controlar los aspectos más importantes de nuestro cluster.

El menú del programa contendrá 4 opciones principales:

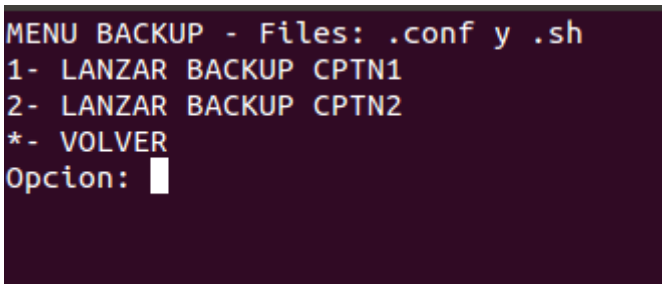
Dentro de nuestro script vamos a poder ver los estados de nuestros servicios para poder controlar en todo momento si están parados o están corriendo correctamente:



También podremos arrancar, reiniciar o parar un determinado servicio de nuestro cluster:



Otra opción que tenemos es la de lanzar un backup de archivos importantes, nosotros lo hemos configurado para que haga una copia de seguridad de los archivos de configuración de nuestros servicios clusterizados, es decir ftp y servidor apache:



A continuación pondremos alguna parte del código que nos parece relevante para ver como realizamos el backup de los archivos y como vemos el estado de los servicios.

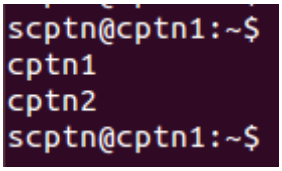
Primeramente mostraremos la parte de backup de servicios de los archivos de configuración del ftp y del Apache, hemos programado un shell script y mediante el comando scp vamos guardando los datos que necesitamos:

```
if [ $opc3 -eq 1 ];then      #### COPIAR ARCHIVOS CONF SERVICIOS CPTN1 Y CPTN2
  echo  Archivos a copiar
  echo /etc/apache2/sites-enabled/000-default.conf
  echo /etc/vsftpd.conf
  scp cptn1:/etc/apache2/sites-enabled/000-default.conf /home/scptn/backup/lastcopy/apachec1.$fecha.conf
  scp cptn1:/etc/vsftpd.conf /home/scptn/backup/lastcopy/vsftpdcp1.$fecha.conf
  echo TERMINADO CPTN1
  read
elif [ $opc3 -eq 2 ];then
  echo Archivos y dir a copiar
  echo /etc/apache2/sites-enabled/000-default.conf
  echo /etc/proftpd/proftpd.conf
  echo /home/scptn/scripts/
  cp /etc/apache2/sites-enabled/000-default.conf /home/scptn/backup/lastcopy/apachec2.$fecha.conf
  cp /etc/proftpd/proftpd.conf /home/scptn/backup/lastcopy/proftpd.$fecha.conf
  cp /home/scptn/scripts/* /home/scptn/backup/lastcopy/scripts/
  echo TERMINADO CPTN2
  read
else
 opc3=3
fi
done
}
```

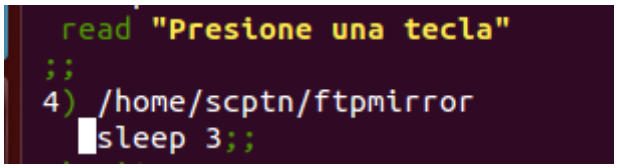
Ahora mostraremos la parte de estado de los servicios, mediante un shh a ambas máquinas y buscando con un grep, cogemos el trozo de información que necesitamos para verificar el estado del servicio FTP y Apahce:

```
1)clear    #VER ESTADO DE SERVICIOS
  echo "1)-CPTN1  2)-CPTN1 3)-CLUSTER *)VOLVER"
  read -p "OPCION:" opc1
  if [ $opc1 -eq 2 ]; then
   echo APACHE - CPTN2
   ssh cptn2 service apache2 status |grep Active
   echo FTP SERVER - CPTN2
   ssh cptn2 service vsftpd status |grep Active
   read
  elif [ $opc1 -eq 1 ]; then
   echo APACHE - CPTN1
   service apache2 status |grep Active
   echo FTP SERVER - CPTN2
   service vsftpd status |grep Active
   read
  elif [ $opc1 -eq 3 ]; then
   echo APACHE - CPTN2
   ssh cptn2 service apache2 status |grep Active
   echo FTP SERVER - CPTN2
   ssh cptn2 service proftpd status |grep Active
   echo APACHE - CPTN1
   service apache2 status |grep Active
   echo FTP SERVER - CPTN1
   service vsftpd status |grep Active
```

También tendremos la posibilidad con la opción "Estado del Cluster", ver si nuestro cluster está funcionando ejecutando el comando *"mpiexec -f /home/scptn/machinefile -n 2 hostname"* de manera automática gracias al script:



Y ya para finalizar, tenemos la opción de lanzar el mirror del ftp para que ambas máquinas tengan la misma información guardada en el disco duro, lo que hacemos es ejecutar nuestro script de lftp, anteriormente explicado en el punto 3, el código es simple y quedaría de la siguiente manera:



4. Servidor ecológico

Montaje y configuración de la parte eléctrica

Nuestro producto está concebido para aquellos sistemas que se puedan automatizar de forma que no se requiera la presencia humana en ningún aspecto.

Aplicaciones prácticas y funcionales, serian control de estación meteorológica, control y monitorización en campos de cultivos agrícolas, control y monitorización de campos eólicos, integración en satélites de comunicación o cualquier tipo y un largo etc.

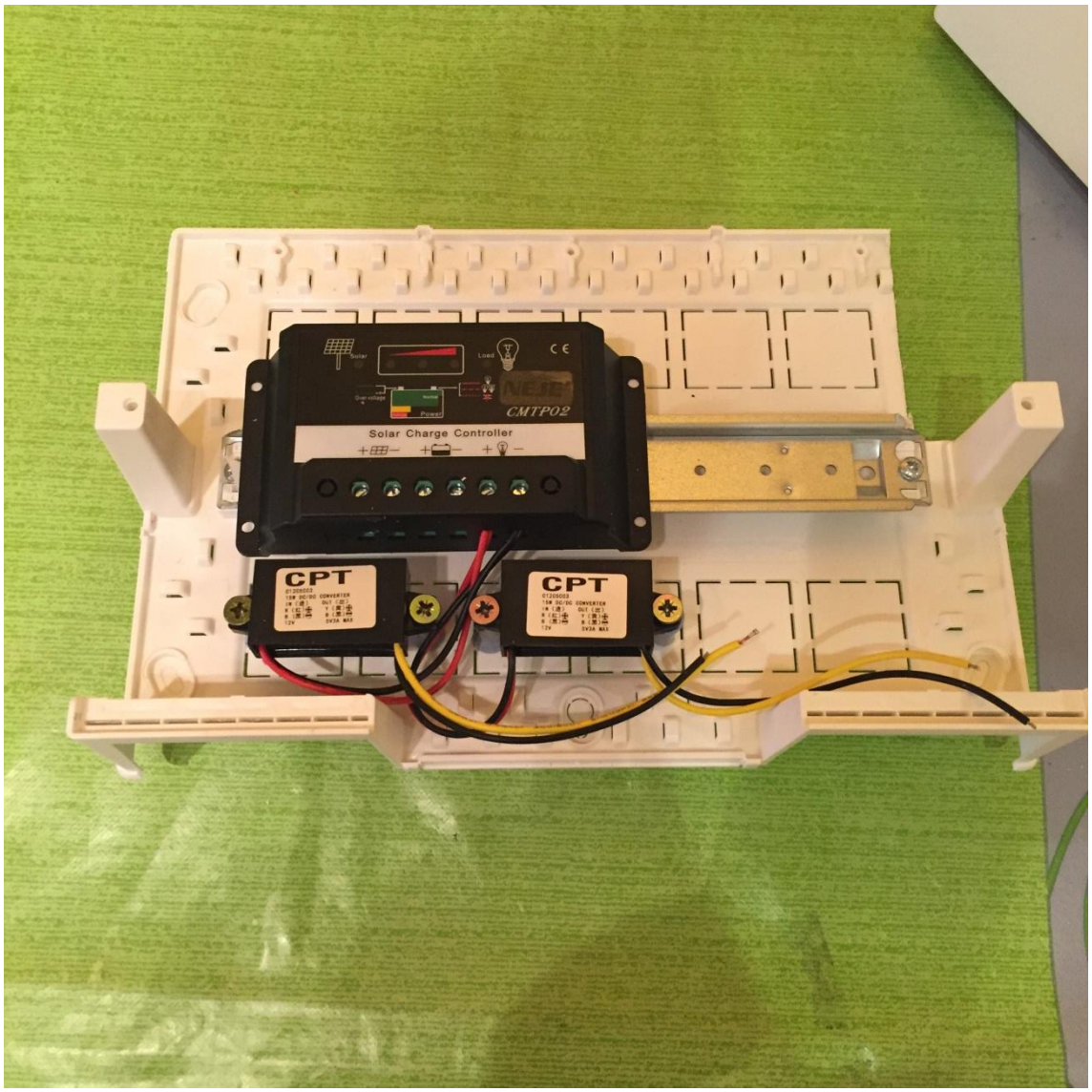
La versatilidad de este producto puede variar en función de las necesidades que dispongamos, ya sea en capacidad de procesamiento, aumentando el cluster de raspberry, en optimización de energía aumentando la superficie de placas solares o en aumento de rendimiento energético y autonomía aumentando la capacidad de almacenamiento de las baterías.

En función del cuadro de necesidades hay un amplio campo de mejoras en el hardware como de implementaciones en el software.

ESPECIFICACIONES TÉCNICAS

Este sistema autónomo está proyectado para funcionar con energía solar o cualquier corriente de entrada a 12v en continua. Preferiblemente una energía renovable que es la finalidad de este proyecto. La tensión de entrada admitiría un rango entre 12v y 30v.

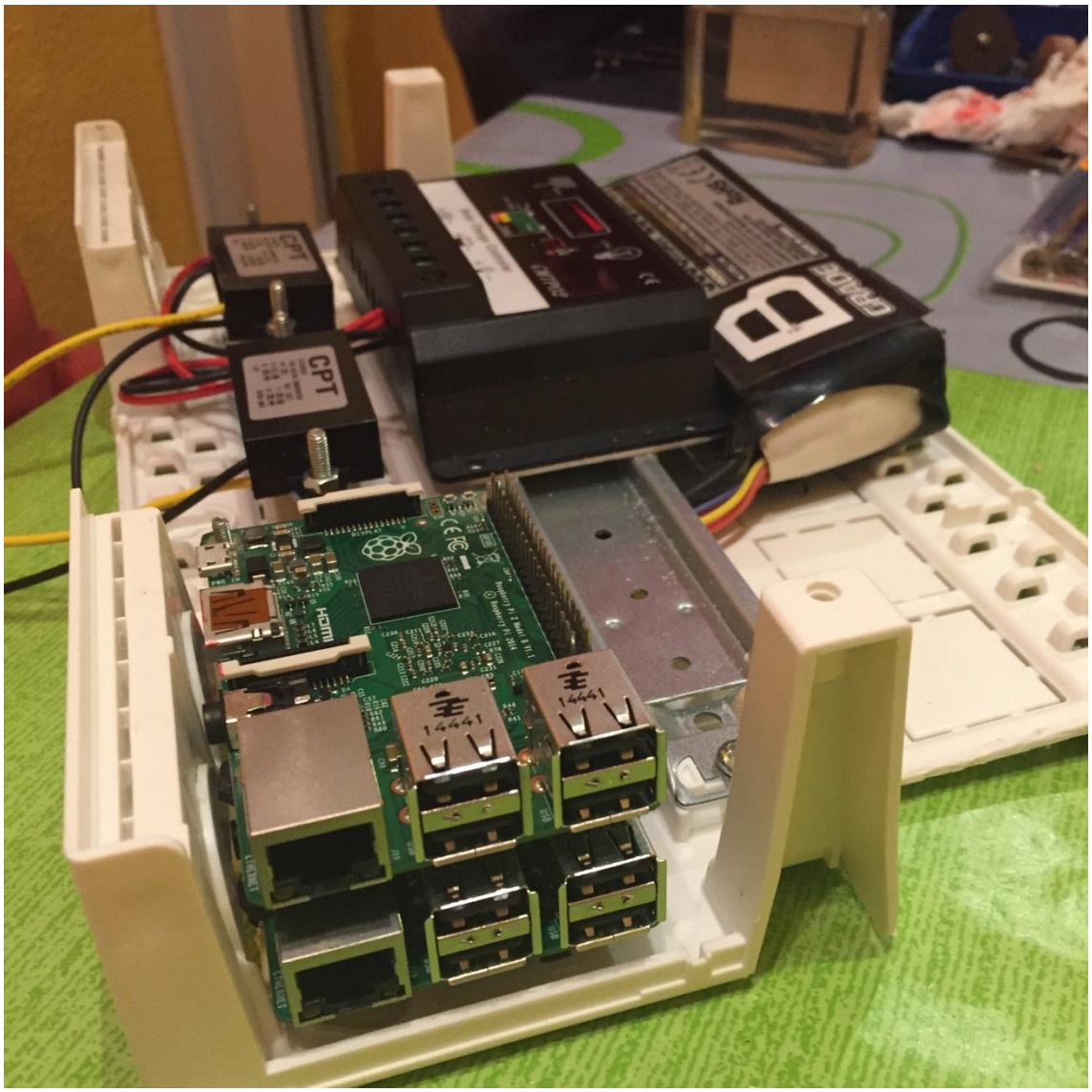
A continuación de la entrada de energía tenemos nuestro gestionador de carga solar REEJE, que realiza la gestión entre : entrada de energía, almacenamiento y suministro al sistema. A su vez realiza una primera transformación dando una salida de 12v en continua que después adaptaremos a nuestras necesidades. En este mismo dispositivo podemos monitorizar si hay una tensión de entrada, el estado de carga de las baterías y si hay un consumo de energía sobre las mismas.



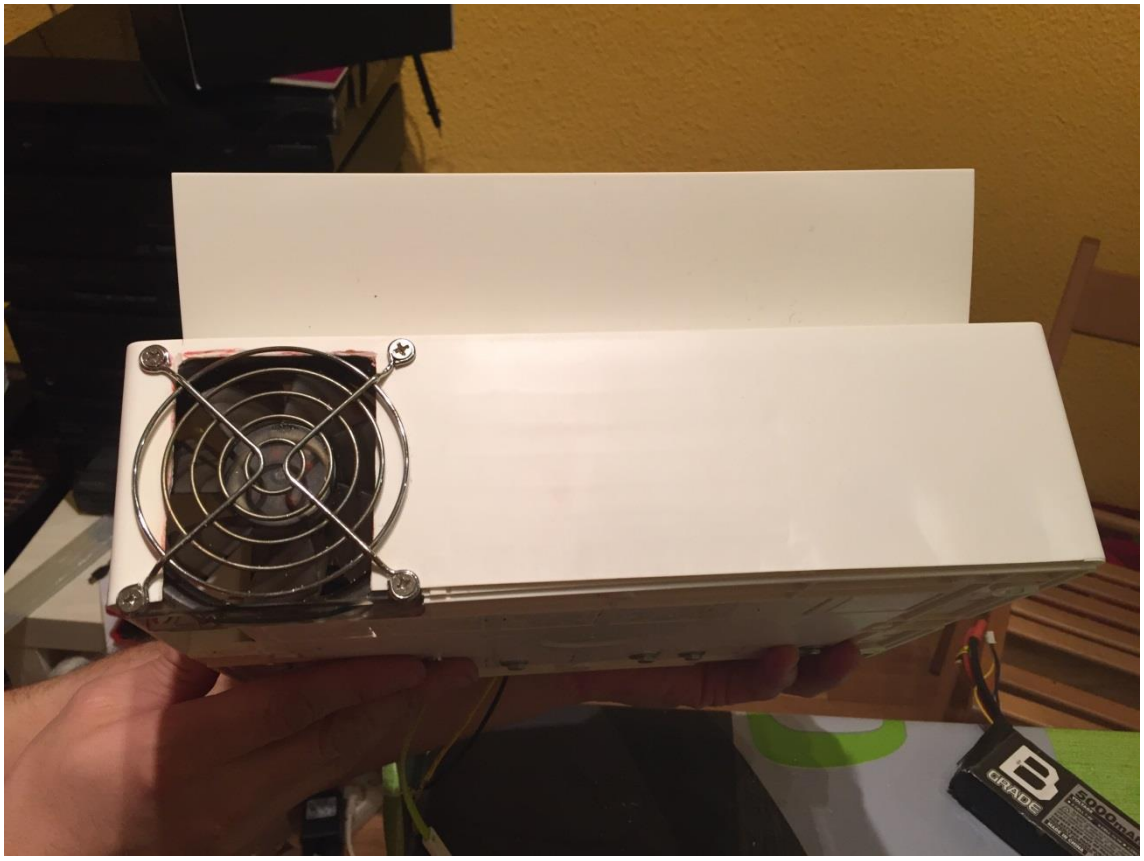
En una segunda etapa de transformación de la tensión hemos optado por 2 transformadores de entrada 12v y salida 5v y un max de 3 amp, que realizan la función de alimentación del cluster y de una seguridad de forma individual a cada nodo. Protegiéndolo de picos de tensión hasta 3 amp que es el max pico de intensidad requerido por cada nodo.

Se ha incluido un amperímetro analógico, para comprobar de forma instantánea la corriente de entrada en las baterías de una capacidad de 0.5 amperios.

Este caso hemos optado por baterías tipo Lipo (litio – polímero) por un gran rendimiento y en poco espacio, buscando un modelo compacto y funcional para esta presentación. Se puede ampliar la capacidad simplemente colocando otro modulo igual de 12v y 5 amp en paralelo obteniendo asi 10 amp de capacidad de almacenamiento.


CPT
CPT
CRAIOVE
HDMI
14441
14441

El sistema refrigeración formado por un ventilador de 12 cm a 12v con un consumo max de 0,2 amp. Regulando por un potenciómetro para calibrar el sistema según su rendimiento.



5. Posibles mejoras

Mejoras de parte eléctrica

Aunque en nuestra maqueta no este implementado dado nuestro presupuesto limitado,para llegar a tener un server cluster real y coompletamente autonomo sería necesario sustituir la batería por una con más capacidad, ya que debe ser lo suficientemente grande para mantener al server funcionando por la noche sin depender de las placas ya que no generarán ninguna energía. Al igual pasaría en dias especialmente nublados, que no podremos lograr recibir ninguna energía del sol y nuestro server tiene que ser capaz de seguir funcionando.

Otro elemento que en nuestra maqueta no se ajusta a los valores reales es nuestra placa solar, de reducidas dimensiones pero que logramos hacer funcionar en simulación recibiendo una carga de 2V, pero para neustro server la mínima carga sería de 5v para alimentar las Raspberry y necesitando una carga mayor para lograr cargar las baterías cuando sea necesario. Esta mejora no cambiaría demasiado el precio siendo totalmente necesaria para el correcto funcionamiento del sistema de energía.

Placa solar auto-dirigible - Se podría reducir el tamaño de las placas solares si somos capaces de que se dirijan solas al punto más eficiente de transformación de energía. Esto se podría conseguir con una base para las placas que podría direccionarse con un servo auto alimentado por nuestra placa y unos sensores de luz que mediante una triangulación calculen el punto de luz más alto.

Mejoras parte informática

Nuestro sistema de cluster está basado en la nivelación de la carga que tienen los servicios entre los nodos Raspberry pi 2 existentes.

Así bien nuestros servicios se repartirán peticiones y harán cálculos de procesos repartiendo la carga de la manera más equitativa posible.

Una mejora que se podría desarrollar en nuestros cluster podría ser la de instalar el veritas cluster server, un software de renombre de la aclamada veritas que podría hacer que implementásemos de manera mucho más eficaz cualquier servicio en nuestros servidores, ya que hasta el momento nos valemos de investigaciones que no en todos los casos llegan a buen puerto.

Para hacer de nuestro server un servicio totalmente autónomo, que aunque ya desde un principio estaba pensado no está instalado de momento deberíamos implementar un sistema 3g o 4g con el que tendremos comunicación continua con el servidor estemos donde estemos mediante una vpn.

Con este método lograremos tener un control de nuestro server en todo momento y tendremos al alcance los datos que necesitemos así como la configuración de nuestro cluster. Esto es esencial para que nuestro server sea utilizado en cualquier tipo de terreno, sin necesidad de energía y sin una conexión cable a la red.

Para estos ambientes hostiles para cualquier tipo de máquina de computación tenemos pensado también añadir una web cam al servidorestratégicamente colocada, al cual mediante un servidor de web cam podamos conectarnos desde el 3 o 4g podamos ver el estado físico de nuestro server en todo momento, conexiones, terreno etc …