Pontificia Universidad Javeriana

Cali


# Sistemas Operativos

John A. Sanabria

Juan Diego Montoya

# Introducción

El objetivo específico de este proyecto es trabajar con el sistema Raspberry Pi para monitorear en tiempo real el estado de algunos de sus componentes. Nos familiarizaremos con el ambiente de cómputo, utilizando el sistema operativo Raspbian Jessie. Mediante la creación de un archivo bash (.sh), realizaremos la consulta de dichos datos para después mediante un script en Python, que más adelante revisaremos de forma más profunda, realizaremos el envío de los datos hacia una plataforma web que realizará las diferentes gráficas. El sistema realizará el monitoreo cada 15 minutos y enviará los datos a la plataforma web Thingspeak.

# Mecanismos de consulta

Para desarrollar el script que permite el monitoreo de los diferentes elementos, primero se identificaron los diferentes comandos o consultas que nos permitirán tener acceso a la información necesitada para realizar nuestro monitoreo periódico. Los diferentes elementos a monitorear, junto con el mecanismo identificado para realizar la consulta de su estado son:

1. CPU (%): El monitoreo de la CPU se realizará mediante el uso del comando “uptime” el cual nos retornará principalmente 3 valores, los cuales hacen referencia a la carga de la CPU en el último minuto, los últimos 5 minutos y los últimos 15 minutos. Mediante el uso del comando cut , tanto para este como para todos los “Split” que haremos en los diferentes comandos, conseguiremos separar únicamente el valor que nos interesa. En este caso nos interesa el tercer valor que es la carga promedio de CPU en los últimos 15 minutos. Si usara otra medición, y el script se ejecuta cada 15 minutos, en verdad tendría los datos del último minuto o últimos 5 minutos, pero perdería rastro de los demás minutos que pasaron en ese intervalo.

2. Memoria (%): El porcentaje de memoria utilizado lo calcularemos a partir de dos datos que consultaremos. Necesitaremos el total de memoria con el que cuenta nuestro ambiente de cómputo y la cantidad de memoria libre. Con estos dos datos mediante regla de 3 sacaremos el porcentaje de memoria utilizado ((Memoria libre * 100) / Total de memoria). Dichos valores los obtendremos del comando "cat /proc/meminfo".

3. Disponibilidad de Disco (%): La disponibilidad de disco la obtendremos del comando "df", el cual nos arroja el porcentaje de disco disponible con el que contamos.

4. Ancho de Banda (Descarga y Carga): El cálculo del ancho de banda, tanto de descarga como de carga es de la misma manera. Para averiguar la velocidad de carga y descarga con el que contamos en la red en la que estemos disponibles, utilizaremos el comando "sudo speedtest-cli". Speedtest-cli nos permite saber el ancho de banda mediante la línea de comandos sin necesidad de contar con una GUI. Para esto, debemos realizar la instalación previamente de dicho comando de la siguiente manera: "sudo pip install speedtest-cli".

5. Temperatura (C°): La temperatura que tiene actualmente nuestra Raspberry Pi la sabremos mediante el comando "/opt/vc/bin/vcgencmd measure_temp" el cual nos entregará la temperatura en grados Celcius (C°).

Los comandos presentados anteriormente nos generarán un formato de respuesta, el cual contiene los valores que estamos buscando pero debemos filtrar nuestro resultado para obtener únicamente el valor deseado. Como se mencionó anteriormente esta partición de cadenas se realizará mediante el uso del comando "cut". Dentro de nuestro script también podemos observar unos "echo" que se utilizan para imprimir algún texto en pantalla durante la ejecución de nuestro script. Éstas marcas no son únicamente para las pruebas realizadas (verificar que los datos que se envían son los mismos que los que se grafican en nuestra plataforma web), sino que también tiene la función de dejar plasmados dichos datos enviados, junto con la fecha y hora de ejecución, en nuestro archivo "output.log". Este log quedará almacenado en la misma carpeta que nuestro

script (se configura manualmente su ruta) y nos permitirá llevar un rastro de los datos que se han enviado. La generación de dicho archivo se explicará de forma detallada en el apartado de la planificación de la tarea (cron job).

# Mecanismos de envío

Para realizar el envío de datos desde nuestra Raspberry Pi hacia nuestra plataforma web, que para este caso utilizaremos la plataforma Thingspeak (https://thingspeak.com/) la cual recibirá los datos que queremos que grafique mediante un envío de webservices utilizando el método POST, utilizaremos un script creado en Python (webServices.py) el cual tiene como parámetros de entrada los 6 elementos que queremos graficar. Por lo cual una vez nuestro script en Bash tenga recolectada la información solicitada, realizará una ejecución del script en Python, al cual se le pasarán dicha información, y nuestro script mediante nuestro API KEY, le enviará los datos solicitados a la plataforma para que ésta los grafique. Tanto éste como el script principal se encontrarán al final del presente documento y adjuntados como se solicitó en los objetivos del proyecto.

# Programación de la tarea

Para la ejecución periódica de nuestro script, he decido realizar un "cron job". Para explicar brevemente qué es un cron job, citaré la información obtenida del sitio web: http://blog.desdelinux.net/cron-crontab-explicados/ , el cual nos explica qué es un cron.

> "¿Qué es cron?
>
> El nombre cron viene del griego chronos que significa "tiempo".
> En el sistema operativo Unix, cron es un administrador regular de

> procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab."

Mediante el comando "crontab –e" modificaremos nuestro administrador de trabajos en cron, para después agregar la ejecución periódica de nuestro script en bash. La manera en la que programamos una tarea en nuestro cron es agregando una línea en nuestro administrador con el siguiente formato:

```
m h dom mon dow user command
```

Dónde:

- **m** : Corresponde al minuto en que se va a ejecutar el script, el valor va de 0 a 59.
- **h** : La hora exacta, se maneja el formato de 24 horas, los valores van de 0 a 23, siendo 0 las 12:00 de la medianoche.
- **dom** : Hace referencia al día del mes, por ejemplo se puede especificar 15 si se quiere ejecutar cada dia 15.
- **dow** : significa el día de la semana, puede ser numérico (0 a 7, donde 0 y 7 son domingo) o las 3 primeras letras del día en inglés: mon, tue, wed, thu, fri, sat, sun.
- **user** : define el usuario que va a ejecutar el comando, puede ser root, u otro usuario diferente siempre y cuando tenga permisos de ejecución del script.
- **command** : refiere al comando o a la ruta absoluta del script a ejecutar, ejemplo:/home/usuario/scripts/actualizar.sh, si acaso llama a un script este debe ser ejecutable.

La línea que se agregó a nuestro administrador para que se ejecute nuestro script cada 15 minutos es:

*/15 * * * * /path/monitor.sh >> /path/output.log

Esto quiere decir que cada 15 minutos, todas las horas, todos los días del mes y todos días de la semana, cualquier usuario que se conecte (en este caso sólo se conectará el

usuario pi), se ejecutará nuestro script llamado monitor.sh y su salida quedará almacenada en un archivo de logs llamado output.log.

# Logs

Como se mencionó anteriormente, cada vez que se ejecute nuestro script, quedará su rastro/trace en nuestro archivo “output”.log, el cual contendrá la información enviada a nuestra plataforma web, el estado del envío y la fecha en la que se realizó la ejecución del script. Cada vez que se ejecute el script, se concatenará a dicho archivo el nuevo registro. El formato que tiene cada registro en nuestro archivo .log es el siguiente:

CPU : 06 %

Memoria : 87 %

Disco : 13 %

Descarga : 3.94 Mbit/s

Carga :  0.29 Mbit/s

Temperatura : 50 Cº

Estado envìo : OK

Sun 28 Feb 06:41:19 UTC 2016

----------FIN----------

# Scripts

A continuación podremos ver los dos scripts con los cuales realizaremos las tareas solicitadas. Uno siendo nuestro script en bash y el otro nuestro script en Python para el envío de datos.

***Monitor.sh:***

```
#!/bin/bash

mem=`cat /proc/meminfo`

memTotal=`echo ${mem} | cut -d ':' -f 2 | cut -d ' ' -f 2`

memFree=`echo ${mem} | cut -d ':' -f 3 | cut -d ' ' -f 2`

let memUsed=(${memFree}\*100)/${memTotal}

let memUsed=100-${memUsed}

disk=`df`

disk=`echo ${disk} | cut -d '%' -f 2 | cut -d ' ' -f 8`

temp=`/opt/vc/bin/vcgencmd measure_temp`

temp=`echo ${temp} | cut -d '=' -f 2`

temp=${temp::-4}

cpu=`uptime`

cpu=`echo ${cpu} | cut -d 'l' -f 2 | cut -d ' ' -f 5`

cpu=${cpu:2:-1}

broadband=`sudo speedtest-cli`

download=`echo ${broadband} | cut -d ':' -f 3 | cut -d 'M' -f 1`

download=${download:1:-1}

upload=`echo ${broadband} | cut -d ':' -f 4 | cut -d 'M' -f 1`

send=`sudo python /home/pi/Documents/Sisoper/Raspberry/webServices.py ${cpu} ${memUsed} ${disk} ${download} ${upload} ${temp}`

echo "CPU : "${cpu} "%"

echo "Memoria : "${memUsed} "%"

echo "Disco : "${disk} "%"

echo "Descarga : "${download} "Mbit/s"

echo "Carga : "${upload} "Mbit/s"

echo "Temperatura : "${temp} "Cº"

echo "Estado envìo : "${send}

echo `date`

echo "------------------------------------------FIN------------------------------------------"
```

***webServices.py***

```
import httplib,urllib

import sys

def push():

   if(len(sys.argv) == 7):

      params = urllib.urlencode({'field1': sys.argv[1], 'field2': sys.argv[2], 'field3': sys.argv[3],'field4': sys.argv[4],'field5': sys.argv[5],'field6': sys.argv[6], 'key':'L7ESGOM5GYFVZ3Z6'})

      headers = {"Content-tupe": "application/x-www-form-urlencoded","Accept": "text/plain"}

      conn = httplib.HTTPConnection("api.thingspeak.com:80")

      try:

            conn.request("POST", "/update", params, headers)

            response = conn.getresponse()

            print response.reason

            data = response.read()

            conn.close()

      except:

            print "Connection Failed"

   else:

      print "6 Parameters Required"

if __name__ == "__main__":

   push()
```