Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

cpudyn: controlar la velocidad de la CPU automáticamente (60636 lectures)

Per **Ricardo Galli Granada**, ***gallir*** (http://mnm.uib.es/gallir/)

Creado el 01/05/2003 16:49 modificado el 16/07/2003 22:51

El cpudyn[1] es un daemon que controla dinámicamente la frecuencia del procesador dependiendo de la carga. El objetivo al desarrollarlo fue tener un programa sencillo que pudiese ser usado siempre (con o sin baterías) en un portátil para bajar el consumo y la temperatura de la CPU, pero afectando lo mínimo posible al rendimiento, especialmente al interactivo.
***Actualización**: Liberada versión 0.4.0, ya funciona[2] con la interfaz estándar (y nueva) cpufreq del 2.6.0-test1.*

cpudyn: controlar la velocidad de la CPU automáticamente

Página del proyecto[1]
Página en Freshmeat[3]

El cpudyn[1] es un daemon que controla dinámicamente la frecuencia del procesador dependiendo de la carga. El objetivo al desarrollarlo fue tener un programa sencillo que pudiese ser usado siempre (con o sin baterías) en un portátil para bajar el consumo y la temepratura de la CPU, pero afectando lo mínimo posible al rendimiento, especialmente al interactivo.

La mayoría de los portátiles actuales ya tienen control de la velocidad de CPU. La tecnología más conocida y usada es la Speedstep de Intel que tiene dos frecuencias de trabajo. El kernel 2.5 ya tiene incluido el `cpufreq`, que permite el control de frecuencias de los procesadores, y existe un parche para el 2.4, la versión de Alan Cox 2.4.21-rc1-ac ya lo tiene incluido.

Pero para controlar la velocidad de la CPU hace falta user un programa o "daemon", y existen varios, desde el cpufreq.cc comentado por Switch[4], o el cpufreqd[5]. Ambos tienen varias características buenas (y algunos fallos también), y el cpufreqd[5] es completísimo, pero no hacía lo que yo pretendía, que sea simple, pequeño y que haga sólo una cosa:

> Subir la velocidad de la CPU cuando hace falta, y bajarla cuando ya no lo haga. Además que reaccione rápidamente al uso de programas interactivos.

Con ésto ahorraríamos energía, vida de procesador y temperatura, sin afectar al *rendimiento interactivo*, sobre todo interactivo, del ordenador. Además quería que se comportase igual estuviese o no funcionando con baterías. ¿Porqué hemos de sacrificar velocidad cuando estamos con baterías? ¿Porqué no ponerla al máximo sólo cuando la necesitamos y ahorrar batería cuando no lo necesitemops? (además que lo que más consume en un portatil actual es la pantalla TFT, no el procesador).

El cpudyn

El objetivo era poder tenerlo siempre funcionando en mi portátil, y que molestará lo menos posible, es decir, que me diese la sensación que siempre voy a máxima velocidad, con o sin baterías. Como estaba ya cansado de buscar y no encontraba nada adecuado (¡simple y que haga sólo eso, pero bien!) me puse a desarrollarlo, el resultado es el cpudyn[1]. Está basado en el cpufreq.cc pero simplificado al eliminar las cosas que no hacían falta (entre otras cosas el uso de características no compatibles con C99), y también enormemente modificado en aquellas partes que afectan

directamente al rendimiento del programa o su capacidad de detectar un consumo más elevado de CPU.

Requerimientos

El programa necesite que el cpufreq del kernel esté compilado y en marcha, es decir debe existir el fichero `/proc/cpufreq`. Lo he probado sobre procesadores Speedstep y PowerPC (iBook). En ambos funcionan perfectamente, aunque he notado que se producen pequeños bloqueos en el caso del iBook (yo tengo el modelo más antiguo), eso es un fallo del driver del kernel para ese procesador, o un problema de hardware. Ya lo investigaré, agradeceré que me digáis como funcionan en otro modelos de Apple.

Instalación

Simplemente hay que bajarse el cpudyn.tgz[1], extraer los ficheros y hacer un

```
make install
```

El `make install` instala el daemon en `/usr/sbin` y también un *script* de arranque en `/etc/init.d/`. Ahora basta con hacer:

```
/etc/init.d/cpudyn start
```

Si queréis que se arranque al inicio del sistema deberéis agregar los enlaces correspondientes. Para el caso de Debian ya lo podéis hacer automáticamente:

```
make install-debian
```

Por supuesto, si no os gusta como funciona, podeís hacer el `make unistall` o `make unistall-debian` :-)

Configuración

Los parámetros por defecto que tiene (que lo podéis ver en el script cpudyn o en el "usage" del programa) ya deberían funcionar correctamente, pero los podéis modificar. Hay tres parámetros básicos:

- -i N (N es un número entero) define el periodo de ejecución del daemon en décimas de segundo.
- -p UP DOWN (UP y DOWN son flotantes entre 0.0 y 1.0). UP define la relación "libre:ocupado" para subir la frecuencia del procesador (o pasar a modo "performance"), por ejemplo 0.5 indica que si en el último período de control la CPU estuve un 50% del tiempo libre, hay que subir la frecuencia. DOWN define la misma relación pero para bajar la frecuencia (o pasar a modo "powersave"), 0.95 indica que hay que bajar la frecuencia si la CPU estuvo libre el 95% del tiempo.
- -d Indica que el cpusysd debe ejecutarse en modo daemon.

Verificar el funcionamiento

Para verificar que esté funcionando podéis dejar el ordenador en reposo unos segundos y si hacéis un `cat /proc/cpufreq` deberéis ver como está en modo *powersave*.

```
gallir@minime:~$ cat /proc/cpufreq
          minimum CPU frequency  -  maximum CPU frequency  -  policy
CPU  0       399000 kHz ( 42 %)  -     931000 kHz (100 %)  -  powersave
```

Si hacéis que el procesador esté ocupado, por ejemplo compilando un programa (o ejecutando `yes > /dev/null`), tendréis que ver como pasa a modo *performance* inmediatamente:

```
gallir@minime:~$ cat /proc/cpufreq
          minimum CPU frequency  -  maximum CPU frequency  -  policy
CPU  0       399000 kHz ( 42 %)  -     931000 kHz (100 %)  -  performance
```

Además, si hacéis un `cat /proc/cpuinfo` tendréis que ver como los MHz y *bogomips* también se actualizan con el estado actual de la CPU.

Debug

Si queréis ver como hace esas cosas en "tiempo real", modificad el `cpudynd.h` quitando el comentario de la definición de DEBUG, recompiladlo y luego ejecutado desde la consola **sin** el parámetro `-d`. El programa imprimirá un mensaje por la salida estándard de error cada vez que cambie de estado o que actúe el "decay" (ver a continuación).

Funcionamiento del control de frecuencia

El daemon está en dos estados: *performance* y *powersave*, y cambia la frecuencia de la CPU de acuerdo al estado en que esté, máxima y mínima respectivamente.

- **Powersave**: controla el uso de CPU de acuerdo al período indicado por -i (1 = 1/10 seg por defecto). Si en el último período el tiempo libre de CPU es menor al indicado por el primer parámetro de -p (0.5 por defecto) se pasa al modo *performance* inmediatamente. Cuando el cpudyn se ejecuta en este modo se ejecuta con alta prioridad con el objetivo de "ganar" al o los procesos que han empezado a consumir CPU y así pasar a modo *performance* lo más rápido posible.
- **Performance**: cuando el cpudyn está en este modo verifica el estado de la CPU al doble del período indicado por -i, además el cpudyn se ejecuta con baja prioridad. Lo hace de esta forma para evitar volver al estado *powersave* de forma prematura y además para no "quitar" CPU a los procesos que se están ejecutando (si estamos en este modo es porque seguro que hay procesos en ejecución). Si el tiempo libre de CPU es mayor al especificado por su segundo parámetro, posiblemente pasará a modo powersave, dependiendo del "**decay**" (ver siguiente sección). Si el cpudyn pasa de modo *powersave* a modo *performance*, hace el cambio de la frecuencia y libera inmediatamente al procesador mediante un *sched_yield()*.

Decay

Uno de los efectos que he notado, por ejemplo compilando el kernel, o accediendo a páginas web, es que el cpudyn volvía "prematuramente" al modo powersave porque el consumo de CPU bajaba rápidamente durante un período corto de tiempo, por ejemplo porque está accediendo a disco. Para evitar este efecto de ping-pong lo que hice fue hacer un "retroceso controlado" contando las veces que se cumplen las condiciones de *powersave*.

La condición para pasar de modo *performance* a *powersave* es que durante N períodos consecutivos (el valor del decay, ahora es 5) se cumpla la condición para pasar de performance a powersave. Si antes de los N períodos el consumo de CPU aumenta nuevamente, el contador se re-inicia sin haber cambiado de estado.

Poner los discos en reposo

También estaba cansado de perder tiempo con el *noflushd*, el `hdparm` y los scripts en `/etc/apm/event.d/` para lograr que los discos se pongan en reposo. Es aún más difícil cuando se tienen sistemas de ficheros con journaling[6], que generan operaciones de E/S sobre el diario. Y eso sin contar que muchas veces los discos no responden correctamente a los parámetros del `hdparm -S`.

Así que pensé ¿porqué no hacer que el mismo cpudyn controlase y pusiese a los discos en reposo de la misma manera a como se hace con el `hdparm -y`? Después de pelearme un poco con los argumentos del ioctl, y a la E/S que generan el Ext3 y el XFS, lo he logrado, creo :-).

A partir de la versión 0.2.0, el cpudyn además es capaz de poner un disco en reposo (modo *standby*) si ha pasado un determinado tiempo sin que haya actividad de E/S sobre ese disco. Un par de segundos antes que se haya cumplido el plazo del *timeout*, el cpudyn fuerza el *flush* de la cache y los *buffers*, para evitar que se haga después que el disco se haya puesto en reposo. Si no hay operaciones adicionales a las generadas por el *flush*, el cpudyn pone el disco en *standby* mediante llamadas *ioctl* similares a las que usa el hdparm con el argumento *-y*.

Lo mejor de todo es que dá igual el sistema de ficheros que tengas, ya que trabaja a nivel de dispositivo de bloque, por lo que ya podrás hacer más silencioso tu ordenador, o ahorrar batería del portátil[7] sin necesidad de usar sólo el Ext2.

Esta característica no se habilita por defecto, sólo lo hace si se especifica la opción `-t` **o** `-h`:

- **-t *timeout***: Esta opción habilita el control de discos, se especifica el timeout en segundos. Si pasado ese tiempo no hay actividad, el cpudyn intentará poner el disco en reposo. Primero fuerza un limpiado del buffer-cache, y luego lo pone en modo *standby*. Si no se especifica la opción `-h`, el dispositivo por defecto es `/dev/hda`
- **-h */dev/disco0[,/dev/disco1]...***: Esta opción también activa el control de discos, si `-t` no se especifica, el timeout por defecto es 120. Los dispositivos indicados pueden ser uno o varios, separados por "," y sin espacios entre ellos. Ejemplo: `cpudynd -t 60 -h /dev/hda,/dev/hdc`

IMPORTANTE: por supuesto que el cpudyn no puede evitar que otros procesos escriban al disco y eviten que se ponga en reposo. Si quieres ver las operaciones que se hacen en cada disco, quita el comentatiro a la definición de DEBUG en el `cpudynd.h`, recompila y arráncalo, te irá mostrando los totales de operaciones de E/S para cada disco.

Comentarios...

Lista de enlaces de este artículo:

1. http://mnm.uib.es/~gallir/cpudyn/
2. http://mnm.uib.es/~gallir/cpudyn/cpudyn/changelog
3. http://freshmeat.net/projects/cpudyn/
4. http://bulma.net/body.phtml?nIdNoticia=1707
5. http://cpufreqd.sourceforge.net/
6. http://bulma.net/body.phtml?nIdNoticia=1153
7. http://bulma.net/body.phtml?nIdNoticia=1510

E-mail del autor: gallir _ARROBA_ uib.es
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1748