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

¿Cómo se calculan los promedios del *loadavg* en el Linux? (12065 lectures)

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

Creado el 09/03/2001 00:00 modificado el 09/03/2001 00:00

> ***Pregunta:***[1] *En diversas ocasiones he oido hablar del comando w, y de que es utilizado para mirar la carga de un ordenador. Me gustaría saber que significan los tres números esos del final de la primera linea. He leido que son la carga media durante los últimos 1, 5 y 15 minutos, pero querría saber en que escala está. Supongo que como más grandes sean los números, más carga tendrás.*

Los números que muestra el *w* o el *uptime*, como bien dices, son los promedios (aproximados) exponenciales de la "carga". La carga es sencillamente el número de procesos en la cola de ***listos para ejecutar*** del kernel (*active_tasks*).

Si tienes un solo proceso haciendo sólo cálculo, el valor de la carga será uno. Básicamente te dá una idea del promedio de procesos compitiendo por CPU. En la mayoría de los casos, prácticamente todos los procesos están bloqueados por entrada/salida, por lo que las cargas serán muy bajas (ver también respuesta[2] de Guillem Cantallops).

Forma de cálculo

La formula para calcular el valor actual de la carga es:

h = Historia (y promedio)
x = Último valor

h = *h* * EXP + (1 - EXP) * *x*

Para calcular el EXP a usar para 1, 5 y 15 minutos se hace:

1 minuto

EXP_1 = 1 / exp(5 seg/1 min) = 1/exp(1/12) =~ 0.91

5 minutos

EXP_5 = 1 / exp(5 seg/5 min) = 1/exp(1/60) =~ 0.98

15 minutos

EXP_15 = 1 / exp(5 seg/15 min) = 1/exp(1/180) =~ 0.99

Se observa que cuanto más grande es EXP, la historia "permanece" por mas tiempo. Se puede decir que la historia tiene un "valor de decaimiento" (*decay*) de 0.91 para el primer caso, 0.98 para el segundo y 0.99 para el tercero.

Evolución de los promedios

Supongamos que en *h(0)* el valor de la carga fue 1, y que luego todos los valores instantáneos medidos cada 5 segundos fueron 0. Con la evolución del tiempo veremos que pasa para algunos *h(t)*. Recordar que en Linux, cada *t* son 5 segundos.

Tiempo	*t*	1 min	5 min	15 min

0 seg	0	1.00	1.00	1.00
5 seg	1	0.92	0.98	0.98
15 seg	3	0.72	0.95	0.98
1 min	12	0.37	0.82	0.94
3 min	36	0.05	0.54	0.82
5 min	60	0.00	0.36	0.71
15 min	180	0.00	0.04	0.35

Implementación en el Kernel

Estas operaciones en el kernel se hacen con aritmética entera, por lo que el código es un poco más complejo.

La función que calcula en el kernel está en *linux/kernel/timer.c*.

linux/kernel/timer.c

```
// Se llama desde update_times(void)
static inline void calc_load(unsigned long ticks)
{
    ...
    // Se ejecuta cada 5 segundos
        count += LOAD_FREQ;
        active_tasks = count_active_tasks();
        CALC_LOAD(avenrun[0], EXP_1, active_tasks);
        CALC_LOAD(avenrun[1], EXP_5, active_tasks);
        CALC_LOAD(avenrun[2], EXP_15, active_tasks);
    }
}
```

pero necesita de las definiciones en *linux/include/linux/sched.h*.

linux/include/linux/sched.h

```
#define FSHIFT      11      /* nr of bits of precision */
#define FIXED_1     (1<<FSHIFT) /* 1.0 as fixed-point */
#define LOAD_FREQ   (5*HZ)      /* 5 sec intervals */
#define EXP_1       1884        /* 1/exp(5sec/1min) as fixed-point */
#define EXP_5       2014        /* 1/exp(5sec/5min) */
#define EXP_15      2037        /* 1/exp(5sec/15min) */

#define CALC_LOAD(load,exp,n) \
    load *= exp; \
    load += n*(FIXED_1-exp); \
    load >&gt= FSHIFT;

// Desplaza el valor anterior en 11 bits a la izquierda
```

La función que imprime los valores de carga en /proc/loadavg está en linux/fs/proc/proc_misc.c

```
static int loadavg_read_proc(char *page, char **start, off_t off,
                 int count, int *eof, void *data)
```

A continuación hay un pequeño programa en C que simula los resultados del *loadavg* para unos pocos intervalos de tiempo

Ejemplo: simloadavg.c

```
// gallir@uib.es, 2001
#include
#define FSHIFT      11      /* nr of bits of precision */
#define FIXED_1     (1<<FSHIFT) /* 1.0 as fixed-point */
#define LOAD_FREQ   (5*HZ)      /* 5 sec intervals */
#define EXP_1       1884        /* 1/exp(5sec/1min) as fixed-point */
```

```
                    // Esto es igual a 1884/2048 = 0.919921...
#define EXP_5       2014        /* 1/exp(5sec/5min) */
#define EXP_15      2037        /* 1/exp(5sec/15min) */


#define CALC_LOAD(load,exp,n) \
    load *= exp; \
    load += n*(FIXED_1-exp); \
    load >>= FSHIFT;

#define LOAD_INT(x) ((x) >&gt FSHIFT)
#define LOAD_FRAC(x) LOAD_INT(((x) & (FIXED_1-1)) * 100)


void imprimir_linea_pf(unsigned long *);

main()
{
        unsigned long a[] = { FIXED_1, FIXED_1, FIXED_1};
                // La carga anterior, 1 en punto fijo
        unsigned long active_tasks = 0;
                // Simulamos que no hay mas carga
        int i, step;

        printf("Exponenciales: %d.%02d %d.%02d %d.%02d\n",
                LOAD_INT(EXP_1), LOAD_FRAC(EXP_1),
        LOAD_INT(EXP_5), LOAD_FRAC(EXP_5),
        LOAD_INT(EXP_15), LOAD_FRAC(EXP_15));

        for(step=1; step
```

Lista de enlaces de este artículo:

1. http://bulma.net/pipermail/bulmailing/2001-March/000312.html
2. http://bulma.net/pipermail/bulmailing/2001-March/000313.html

---

E-mail del autor: gallir _ARROBA_ uib.es
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=550