Always acting for you.
ACT
The Art of Computer Technologies, Corp.

SolarisのCPUロードアベレージについて

株式会社エイ・シー・テイ
2004年1月19日　Rev._

はじめに

性能指標の一つ、CPUロードアベレージについて解説します。

1. コマンド

uptime(1)とw(1)コマンドはCPUのロードアベレージを含む情報を表示します。uptime(1)コマンドの1行はw(1)コマンドの最初の行と同じ内容です。図1に例を示します。なお、本解説では、uptime(1)、w(1)コマンドいずれも同じ表示内容のため、uptime(1)コマンドを中心に解説することにします。

```
# uptime
  9:08pm  up  6:33,  4 users,  load average:   18.29, 7.35, 2.78
# w
  9:08pm  up  6:33,  4 users,  load average:   18.29, 7.35, 2.78
User     tty          login@  idle   JCPU   PCPU  what
oracle   pts/2        2:53pm  1:56   4:35          -sh
nuruki   pts/3        2:53pm  1:53     41          -sh
nuruki   pts/5        6:44pm    17   6:02          -sh
```

図1. uptime(1)コマンドとw(1)コマンドの出力例

本解説ではuptime(1)コマンドで表示される"load average:"についてどのような振る舞いをするのか、またカーネルはどのように値を算出しているのかを述べます。"load average:"の最初の値は1分間、真ん中は5分間、最後は15分間の最近のCPUロードアベレージです。

2. uptime(1)のカーネルインタフェース

uptime(1)コマンドはgetloadavg(3C)ライブラリ関数を用いてカーネルから値を入手し、標準出力に表示します。getloadavg(3C)ライブラリ関数は4.3BSDに最初に実装されました。

```
#include <sys/loadavg.h>
main ()
{
    double  loadavg[3];
    getloadavg(loadavg,3);
    printf("%.2f %.2f %.2f¥n",loadavg[LOADAVG_1MIN],  loadavg[LOADAVG_5MIN], loadavg[LOADAVG_15MIN]);
}
```

図2. getloadavg(3C)の使用例

図3にコマンドからカーネルに至る流れを示します。

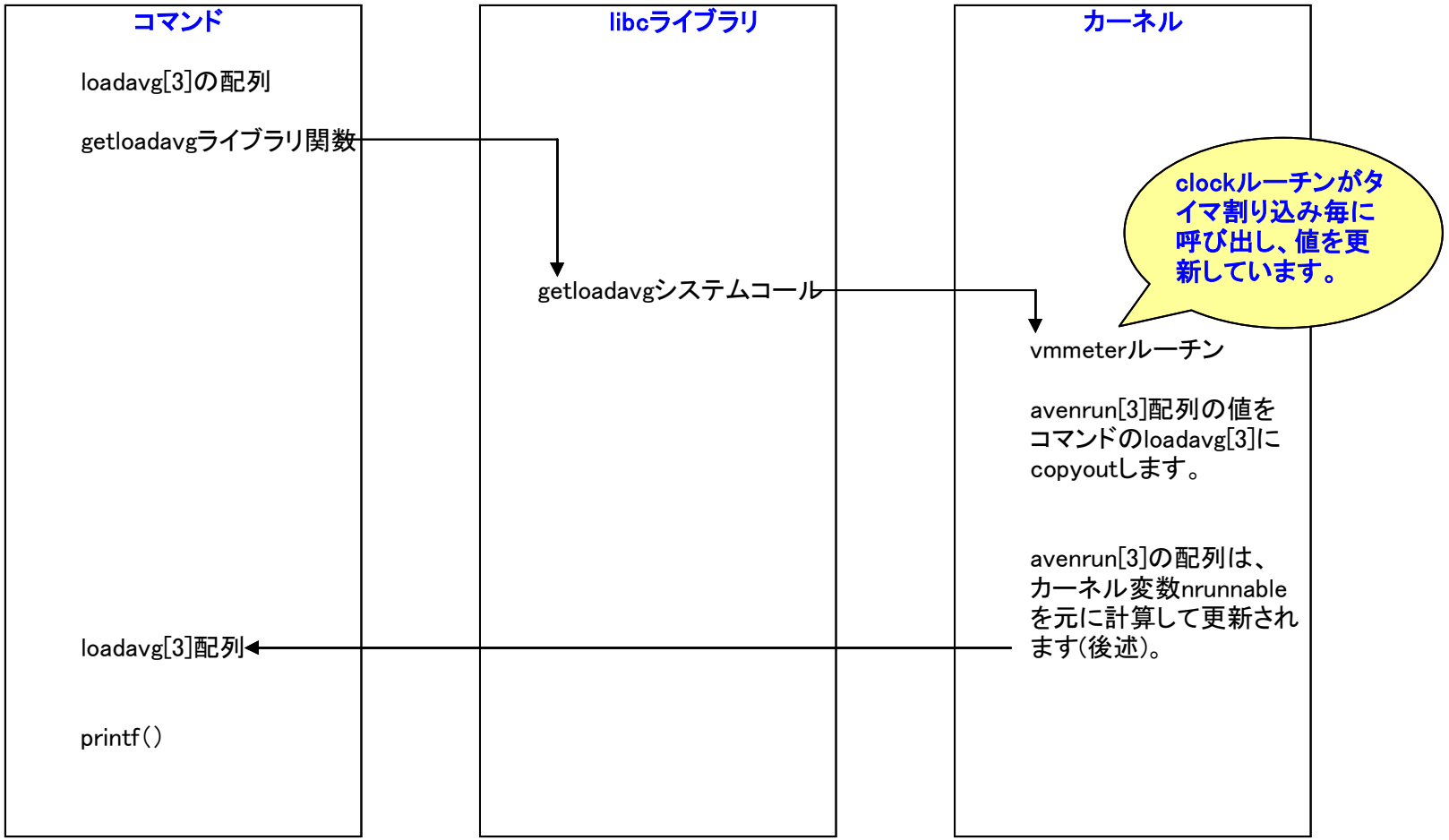

図3. CPUロードアベレージ取得の流れ

図4はカーネルのvmmeterルーチンの要点を抜き出してショートプログラムにしたものです。

```
#include <math.h>
#include <sys/param.h>
#include <sys/loadavg.h>
main (int argc, char *argv[])
{
      int        value;
      double  avenrun[3] = { 0,0,0 };
      double  loadavg[3];
       int      a=-1/60,b=-1/300,c=-1/900;
       int      i,q,r;
       static int x[3] = { 0,0,0 };
       static int f[3] = { 1083,218,73 };          /* for exponential decay */
      value=atoi(argv[1]);
      getloadavg(loadavg,3);
      printf("%12.8f %12.8f %12.8f¥n",loadavg[0],loadavg[1],loadavg[2]);
       a=1-exp(a);
      b=1-exp(b);
      c=1-exp(c);
      a=a << 16;
      b=b << 16;
      c=c << 16;
      printf("%12.8f %12.8f %12.8f¥n",a,b,c);
      for (i=0; i<3; i++) {
              q=x[i] >> 16;
              r=x[i] & 0xffff
              x[i] += (value-q) * f[i] - ((r * f[i]) >> 16);
              avenrun[i] = x[i] >> (16 - FSHIFT);
      }
      printf("%12.8f %12.8f %12.8f¥n",avenrun[0],avenrun[1],avenrun[2]);
}
```

この計算式でavenrun[3]
配列を更新するとともに、
値を算出しています。

図4. CPUロードアベレージのショートプログラム

図4のショートプログラムに5の引数(value はカーネルのnrunnable変数に相当します)を与えて実行すると次のような出力が得られます。

```
# loadavg   5
   0.00390625   0.00781250   0.01171875
   0.00000000   0.00000000   0.01171875
  21.00000000   4.00000000   1.00000000
#
```

図5. ショートプログラムの実行結果

x[3]配列は、カーネル内でhp_avenrun[3]として定義されており、これがavenrun[3]配列を経由してプログラムに渡されます。
引数に指定されるnrunnable変数は、clockルーチンで、実行中、exit中、割り込みで一時停止中等(running)、実行可能なプロセス数(runnable)を合計した値になります。これに、エクスポネンシャル(秒あたりの計算値)を用いて値を求めています。実際はx[3]配列の更新により値が変化するため、ショートプログラムの出力は正しい値を示していないようです。
この関係を明らかにするため、実測データを採取しました。

3. CPUロードアベレージの振る舞い

図6にsar(1)コマンド-uオプションで出力されるCPU使用率と、-qオプションで出力される実行待ちの値を示します。テストでは12個のCPUループするプログラムを同時に実行し、CPUのラッシュ状態を作りました。

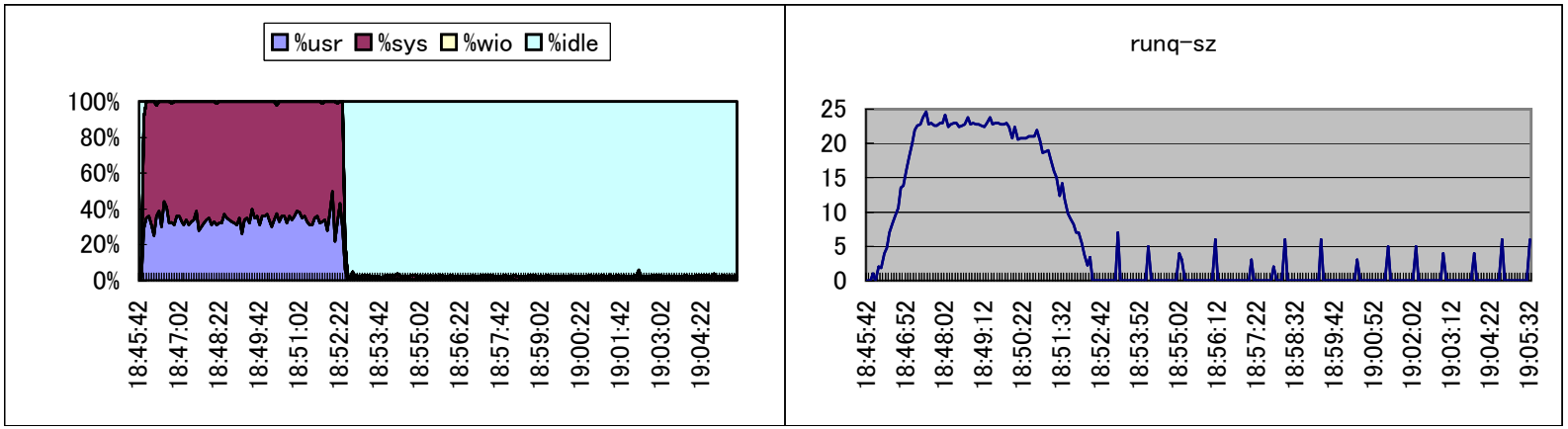

図6. CPU使用率と実行待ち

この時のuptime(1)コマンドの出力をグラフにしたものが図7です。

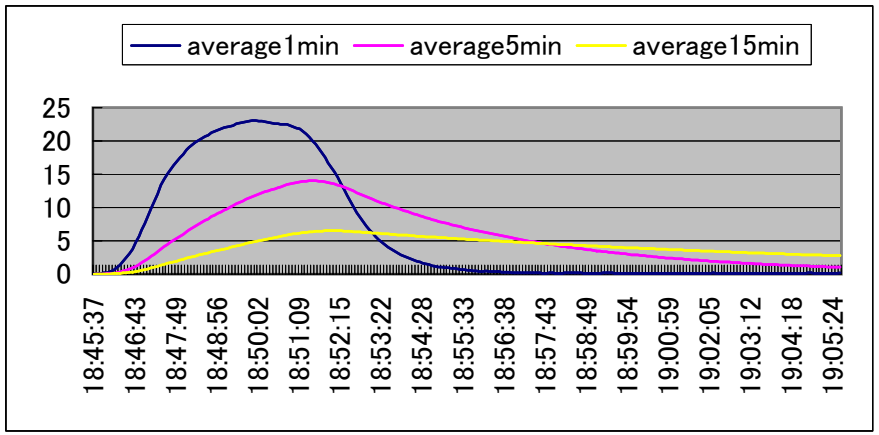

図7. uptime(1)コマンドの出力

この結果、最近の1分間の値がrunq-szのカーブに近いことが判ります。

4. 考察

幾つかのシステム管理ソフトウェアはw(1)コマンドの出力を用いてCPUのロードアベレージを評価しています。以上のテスト結果から、uptime(1)と同様にCPUの負荷を計測するのに妥当な指標であることが明らかになりました。しかしながら、w(1)コマンドのロードアベレージを利用する際、他の項目を用いないのであれば、uptime(1)コマンドがより簡単な出力結果を表示してくれますので、それを利用するほうが良いでしょう。

END REPORT