Ubuntu/LinuxCPUコアのシールド分離

リアルタイム性が要求されるネットワーク通信を含むプログラムなど、さまざまなイベントに応答する際にレイテンシーを最小限に抑える必要がある。 これを行う為に、割り込み (IRQ) とさまざまな CPU 上のユーザープロセスを相互に分離する手法について備忘録を残す。

尚、シールドモデルの考え方など下記の資料が参考になる。

RedHawkLinux_UsersGuide8.2_Jpn.pdf

プロセスのCPUコア分離

リアルタイム性が必要なユーザプログラム（プロセス）を実行するCPUコアを分離する。

コア除外の設定

/etc/default/grubファイルに以下設定をしコア#2,#3をスケジュールから除外。

GRUB_CMDLINE_LINUX_DEFAULT="isolcpus=2,3" （連番のCPU番号指定は2-3と書いてもよい）

```
$ sudo update-grub
$ sudo reboot
```

上記設定は、カーネルの起動引数に isolcpus=2,3 を渡すもの。尚、カーネル起動コマンドに設定されているかの確認は下記。

```
$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-5.15.0-52-generic root=/dev/mapper/ubuntu--vg-ubuntu--lv
ro isolcpus=2,3
```

尚Ubuntu 22.04 Serverでは、何故か？上記では固定出来なかったので、下記コマンドで行った。

```
$ sudo ps -e -o pid= | xargs -n 1 taskset -p 0xFFFFFFF3
```

コア固定の設定

実行するプログラムを、コア#2,#3に固定し実行。

```
$ sudo taskset -c 2-3 <progfile>
```

実行コアの確認

```
$ ps
    PID TTY          TIME CMD
   1067 pts/0    00:00:00 bash
```

```
   2466 pts/0    00:00:00 ps

$ taskset -pc 1067
プロセス ID 1067 の現在の親和性リスト: 0,1
```

割り込みのCPUコア固定

上記でコア分離したプロセスが使用するNICカードの割り込みを、分離したコア側に割り当てる。

IRQ バランスからの CPU の除外

/etc/default/irqbalance ファイルを編集して、コア#2,#3のCPUコアをIRQ バランシングから除外する。

/etc/default/irqbalance

```
IRQBALANCE_BANNED_CPUS=0000000c
```

IRQBALANCE_BANNED_CPUSの指定がなければ、前記のisolcpus=の指定を参照する模様。

NICカードの IRQ への CPU アフィニティーの手動割り当て

/proc/interrupts ファイルを参照して、各デバイスが使用している IRQ を確認する。

```
$ cat /proc/interrupts
            CPU0       CPU1       CPU2       CPU3
   0:         17          0          0          0  IR-IO-APIC    2-edge
timer
   6:          0          0          0          1  IR-IO-APIC    6-edge
ttyS2
   8:          0          0          1          0  IR-IO-APIC    8-edge
rtc0
   9:          0          5          0          0  IR-IO-APIC    9-fasteoi
acpi
 ...
```

該当するIRQ番号専用のsmp_affinityファイルに割り当てるCPUコアのbitマスク値を書き込む。このファイルは、/proc/irq/(IRQ番号)/ ディレクトリにある。
例えばIRQ 番号 130 の割り込みを CPU #2,#3 でのみ実行するよう指示するには下記。

```
$ sudo echo c > /proc/irq/130/smp_affinity
```

尚、設定されたかどうかを、smap_affinity_listファイルを表示して確認する。

```
$ cat /proc/irq/130/smp_affinity_list
```

スクリプトによる IRQ への CPU アフィニティーの自動割り当て

smp_affinityの指定は、再起動した時には元に戻るので、下記のようなPerlスクリプトを実行して自動設定する。

irq_shield.pl

```
#!/usr/bin/perl
#
#指定NICインタフェースの割り込みをコア2,3に割り当てる。
#

use strict;
use warnings;

my $infile = "/proc/interrupts";
my $nicname = "enp3s0";                    #検索するNICインタフェース名
my $mode = 0;                              #動作モード

if (@ARGV >= 1) {
    $nicname = $ARGV[0];
}
if (@ARGV >= 2) {
    if ($ARGV[1] eq "set") {
    $mode = 1;
    }
    if ($ARGV[1] eq "debug") {
    $mode = 2;
    }
}

open(IN, "$infile") || die "can not open $infile\n";

while ( <IN> ) {
    if ( $_ =~ /\b$nicname/ ) {
    my @a = split();
    $a[0] =~ s/:$//;                # IRQ番号、先頭ワードの末尾':'文字を削除

    if ($mode == 0) {
        print;
    }
    else {
        if ($mode == 1) {           # set ?
        system("echo c > /proc/irq/$a[0]/smp_affinity"); #コア2,3
        }
        print "echo c > /proc/irq/$a[0]/smp_affinity\n";
    }
```

```
    }
}

close(IN);
```

上記スクリプトを管理者権限で実行する。

```
$ sudo ./irq_shield.pl enp1s0 set
```

その他

CPUに負荷をかける

stressコマンドで、CPUのコア#2に演算負荷をかける。

```
$ taskset -c 2 stress -c 1
```

CPU負荷状況の確認

下記コマンドでCPUの負荷状況を表示。

```
$ dstat -c -C 0,1,2,3
-----cpu0-usage----------cpu1-usage----------cpu2-usage----------cpu3-usage-
---
usr sys idl wai stl:usr sys idl wai stl:usr sys idl wai stl:usr sys idl wai
stl
  5   4  91   0   0:  7   2  90   0   0: 17   0  83   0   0:  0   0 100   0
0
 11   6  83   0   0:  9   0  91   0   0: 99   1   0   0   0:  1   1  98   0
0
 11   4  85   0   0:  8   0  92   0   0: 98   2   0   0   0:  0   1  99   0
0
  9   4  86   0   0:  5   0  95   0   0: 99   1   0   0   0:  1   1  98   0
0
  9   7  84   0   0:  8   1  91   0   0: 99   1   0   0   0:  0   2  98   0
0
 14   5  81   0   0:  7   0  93   0   0:100   0   0   0   0:  0   1  99   0
0
 12   5  83   0   0:  8   0  92   0   0: 99   1   0   0   0:  0   1  99   0
0
  9   5  86   0   0:  4   1  95   0   0: 99   1   0   0   0:  0   2  98   0
0
 14   6  80   0   0:  5   0  95   0   0: 98   2   0   0   0:  2   0  98   0
0
 10   5  85   0   0:  6   0  94   0   0:100   0   0   0   0:  0   1  99   0
```

```
0
 10   7  83   0   0:  4   0  96   0   0:100   0   0   0   0:  0   0 100   0
0
```

参考

---

1. 第12章 割り込みとユーザープロセスを分離してシステムレイテンシーを最小限に抑える 2. 2.4. 割り込みおよびプロセスバインディング 3. 6 パフォーマンス・チューニング 4. Interrupt Affinityについて 5. 5.3. /proc/配下のディレクトリ 6. irqbalance を NetBSD に移植してみる 7. レッドハットが考えるNFV環境向けのチューニング 8. isolcpusがアクティブ化されているかどうかを検出する方法 9. プロセス・スレッドを特定のCPUコアでのみ動作させる方法 10. SYSTEMD CORE ISOLATION 11. 3.4. irqbalance 12. プロセス・スレッドにCPUアフィニティを設定する方法 13. 第28章 systemd を使用した CPU のアフィニティーおよび NUMA ポリシーの設定 14. 6.3.6.2. CPU の分離 15. stress コマンドを使ってマシンに負荷をかける 16. RedHawkLinux_UsersGuide8.2_Jpn.pdf

From:
https://www.yokoweb.net/dokuwiki/ - **新石器Wiki**

Permanent link:
https://www.yokoweb.net/dokuwiki/programing/linux-programing/linux-cpu-core-shield?rev=1666335999

Last update: **2022/10/21 16:06**