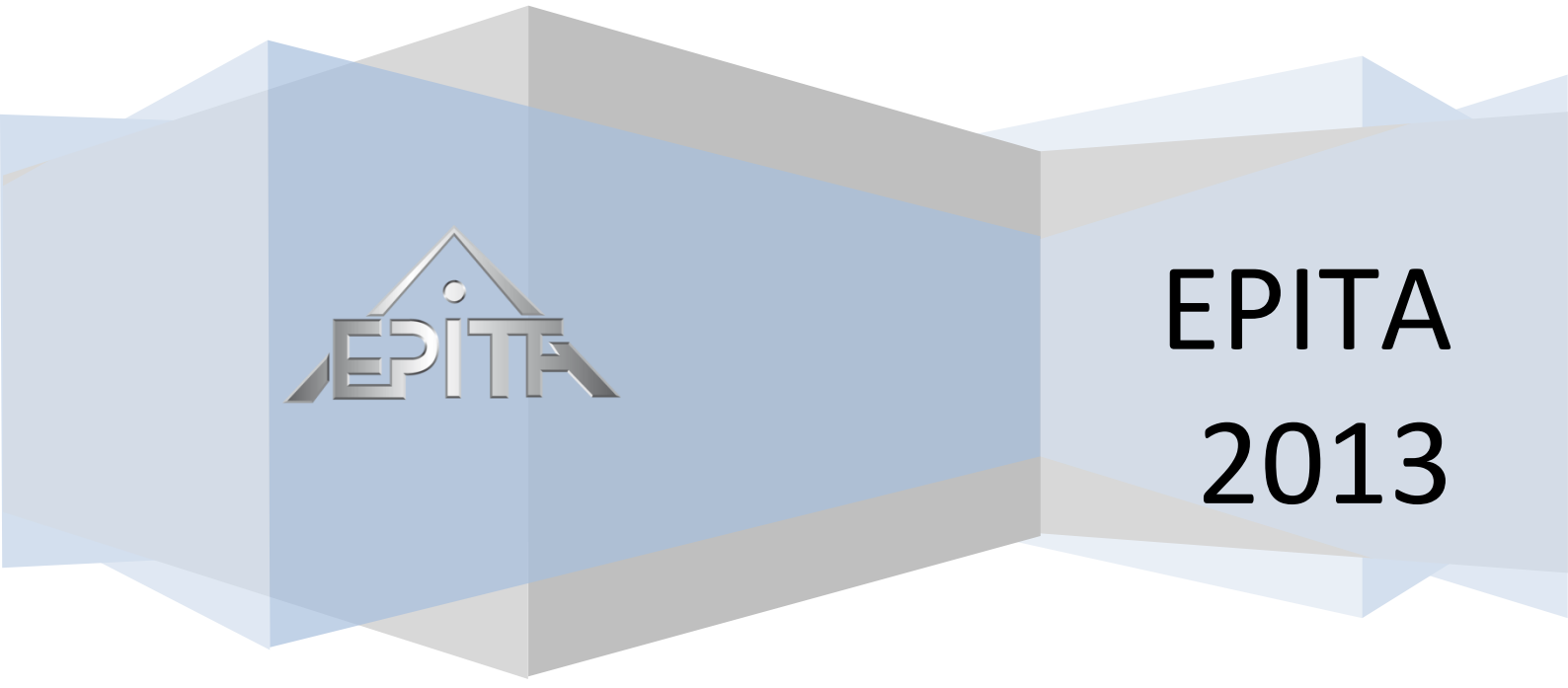
Compte rendu – LIRT

Linux noyau Temps Réel

Rapport

COTARD Philippe
ROLLAND Yves
ZACCARIA Dorian



EPITA
2013

Table des matières

Introduction

Voici le compte rendu de notre travail effectué pour le projet final de LIRT. Ce document comporte toutes nos constatations et calculs de l'ensemble des travaux qui ont été réalisé pour cet exercice.

Tous nos tests on été effectués sur une durée de 30 secondes, le résultat des tests se présente sous la forme d'un tableau montrant une distribution de deltas, le résultat est une courbe de Gauss. Ce delta est en fait le retard que le processeur met pour effectuer le usleep.

Noyaux standard

Sans le ping de « flood »

Le programme est lancé en mode temps réel avec une priorité de 99 sur un Intel I5-2300, la distribution nous montre que la majorité des deltas se situent à 16 µs. Il s'agit du seuil de préemption.

```
gcc -pedantic -ansi -lrt -o preempt preempt.c
sudo chrt --fifo 99 ./preempt
delta = 0 ==> elements = 0
delta = 2 ==> elements = 0
delta = 4 ==> elements = 0
delta = 6 ==> elements = 0
delta = 8 ==> elements = 0
delta = 10 ==> elements = 0
delta = 12 ==> elements = 1
delta = 14 ==> elements = 48
delta = 16 ==> elements = 155
delta = 18 ==> elements = 61
delta = 20 ==> elements = 20
delta = 22 ==> elements = 8
delta = 24 ==> elements = 5
delta = 26 ==> elements = 2
delta = 28 ==> elements = 0
delta = 30 ==> elements = 0
delta = 32 ==> elements = 0
delta = 34 ==> elements = 0
delta = 36 ==> elements = 0
delta = 38 ==> elements = 0
delta = 40 ==> elements = 0
delta = 42 ==> elements = 0
delta = 44 ==> elements = 0
delta = 46 ==> elements = 0
delta = 48 ==> elements = 0
```

Avec le ping de « flood »

Lorsque la machine est soumise à une stimulation continue, le seuil de préemption tombe aux alentours de 6 µs.

```
gcc -pedantic -ansi -lrt -o preempt preempt.c
sudo chrt --fifo 99 ./preempt
delta = 0 ==> elements = 0
delta = 2 ==> elements = 93
delta = 4 ==> elements = 90
delta = 6 ==> elements = 100
delta = 8 ==> elements = 16
delta = 10 ==> elements = 1
delta = 12 ==> elements = 0
delta = 14 ==> elements = 0
delta = 16 ==> elements = 0
delta = 18 ==> elements = 0
```

```
delta = 20 ==> elements = 0
delta = 22 ==> elements = 0
delta = 24 ==> elements = 0
delta = 26 ==> elements = 0
delta = 28 ==> elements = 0
delta = 30 ==> elements = 0
delta = 32 ==> elements = 0
delta = 34 ==> elements = 0
delta = 36 ==> elements = 0
delta = 38 ==> elements = 0
delta = 40 ==> elements = 0
delta = 42 ==> elements = 0
delta = 44 ==> elements = 0
delta = 46 ==> elements = 0
delta = 48 ==> elements = 0
```

Ce phénomène est du au fait que le noyau n'a pas le temps de se mettre en inactivité. Il ne fait donc plus de changements de contexte qui sont couteux en temps. C'est pour cela que les résultats sont meilleurs (toujours en termes de temps).

Noyaux temps réel

La même série de tests a été effectué avec un noyau « real time ».

Sans le ping de « flood »

Le retard du test tombe en moyenne à 10 µs donc inférieur au résultat du même test sur un noyau normal.

```
gcc -pedantic -ansi -lrt -o preempt preempt.c
sudo chrt --fifo 99 ./preempt
delta = 0 ==> elements = 0
delta = 2 ==> elements = 0
delta = 4 ==> elements = 0
delta = 6 ==> elements = 0
delta = 8 ==> elements = 0
delta = 10 ==> elements = 80
delta = 12 ==> elements = 33
delta = 14 ==> elements = 26
delta = 16 ==> elements = 14
delta = 18 ==> elements = 2
delta = 20 ==> elements = 2
delta = 22 ==> elements = 3
delta = 24 ==> elements = 1
delta = 26 ==> elements = 0
delta = 28 ==> elements = 4
delta = 30 ==> elements = 10
delta = 32 ==> elements = 17
delta = 34 ==> elements = 20
delta = 36 ==> elements = 27
delta = 38 ==> elements = 23
delta = 40 ==> elements = 14
delta = 42 ==> elements = 11
```

```
delta = 44 ==> elements = 3
delta = 46 ==> elements = 0
delta = 48 ==> elements = 0
```

On note quand même que la distribution contient beaucoup de valeurs éparpillées.

Avec le ping de « flood »

Lorsqu'on « flood » la machine, le delta retombe à 4 µs pour les mêmes raisons qu'expliquée précédemment.

```
gcc -pedantic -ansi -lrt -o preempt preempt.c
sudo chrt --fifo 99 ./preempt
delta = 0 ==> elements = 0
delta = 2 ==> elements = 44
delta = 4 ==> elements = 120
delta = 6 ==> elements = 100
delta = 8 ==> elements = 12
delta = 10 ==> elements = 6
delta = 12 ==> elements = 7
delta = 14 ==> elements = 7
delta = 16 ==> elements = 3
delta = 18 ==> elements = 0
delta = 20 ==> elements = 0
delta = 22 ==> elements = 1
delta = 24 ==> elements = 0
delta = 26 ==> elements = 0
delta = 28 ==> elements = 0
delta = 30 ==> elements = 0
delta = 32 ==> elements = 0
delta = 34 ==> elements = 0
delta = 36 ==> elements = 0
delta = 38 ==> elements = 0
delta = 40 ==> elements = 0
delta = 42 ==> elements = 0
delta = 44 ==> elements = 0
delta = 46 ==> elements = 0
delta = 48 ==> elements = 0
```

On peut exécuter ce code sur un noyau Xenomai pour avoir de meilleurs résultats, de cette manière le code sera exécuté en priorité par rapport aux à tous les autres threads.