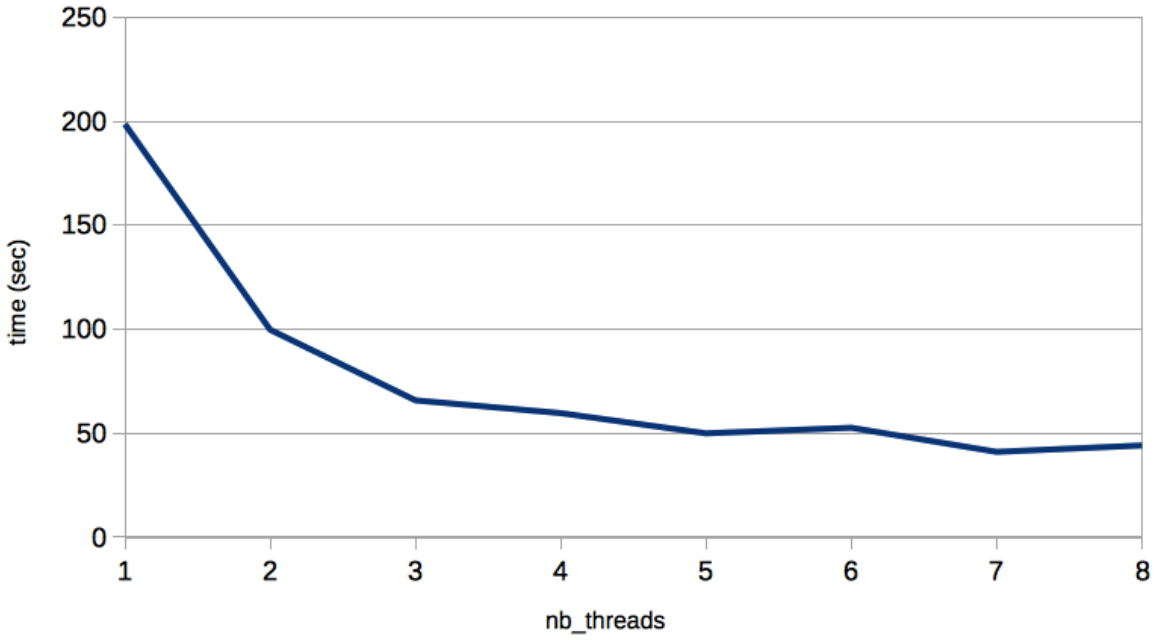
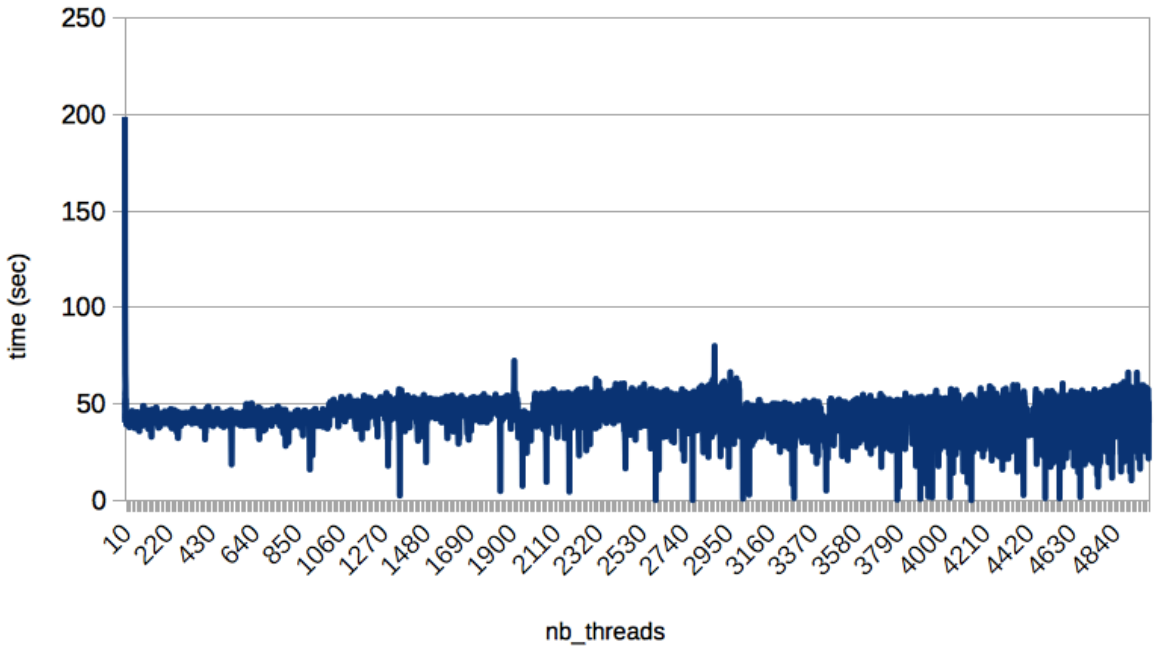
Steven Liatti Orphée Antoniadis Ludovic Gindre

Password Cracker – Groupe 10

Question 1 :





Nous avons ici 2 graphs, le premier montre l'évolution du temps pour trouver le même mot de passe (ici *crypt*) de 1 à 8 threads. Le deuxième quant à lui nous montre la même évolution sur 5000 threads.

Pour nos tests, nous avons utilisé les machines de l'école, pour rappel voici ce que donne `lscpu` sur ces machines :

```
Architecture:          x86_64
Mode(s) opératoire(s) des processeurs :32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) par cœur : 2
Cœur(s) par socket : 4
Socket(s):             1
Nœud(s) NUMA :       1
Identifiant constructeur :GenuineIntel
Famille de processeur :6
Modèle :             60
Révision :           3
Vitesse du processeur en MHz :3940.734
BogoMIPS:              7183.11
Virtualisation :      VT-x
Cache L1d :           32K
Cache L1i :           32K
Cache L2 :            256K
Cache L3 :            8192K
NUMA node0 CPU(s):     0-7
```

Question 2 :

Nous pouvons voir que pour le premier graph, il y a un net gain de temps entre 1 et 4 threads (on passe de 200 secondes pour cracker le mot de passe à 50) puis une stabilisation autours des 50 secondes entre 4 et 8 threads.
Pour le second graph nous pouvons voir la descente observée dans le graph 1 au début du graph puis une stabilisation autours de 50 secondes. On constate tout de même que certaines fois, le mot de passe est trouvé beaucoup plus rapidement (entre 5 et 20 secondes). Ceci est du au fait que le résultat n'est pas déterministe et que sur 5000 tests, nous avons eu parfois plus de chance (le bon thread prenant de l'avance sur les autres).

Le gain de vitesse n'est donc pas linéaire avec le nombre de threads. Il y a effectivement un gain de vitesse de 1 à 4 threads mais la différence n'est plus notable par la suite.

Question 3 :

On constate que l'hyperthreading permet un petit gain de temps (environ 30%) quand on passe de 4 à 8 threads. Ce n'est pas un gain de temps double car deux même threads sur un même core se partagent une seule ALU, il y a tout de même un goulet à ce niveau-là.