Combien de fils d'exécution ?

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 4 septembre 2008

https://www.bortzmeyer.org/combien-de-fils.html

Je viens de commencer un programme comportant plusieurs fils d'éxcution ("*threads*"). Lorsqu'on découpe un programme en fils, il y a une décision à prendre concernant le nombre de fils ; beaucoup de fils permet un parallélisme maximal mais implique aussi un surtravail dû à la communication avec ces fils. Peu de fils fait que du travail peut rester bloqué car le fil en est toujours au travail précédent. La valeur optimale dépend évidemment du problème précis.

Ici, il s'agissait d'un projet AFNIC d'interrogation des zones DNS sous `.fr` pour en déduire des informations, par exemple sur le niveau de déploiement de IPv6 ou bien de DNSSEC (projet DNSwitness <https://www.bortzmeyer.org/dnswitness-expose-ripe.html>). Les requêtes DNS aux serveurs de noms peuvent être très lentes, surtout si un ou plusieurs serveurs de la zone sont en panne, et qu'on doit attendre l'expiration du délai de garde. Avec un programme séquentiel, quelques pourcents de zones vraiment cassées peuvent retarder considérablement tout le programme. D'où l'idée de parallèliser.

`.fr` compte actuellement environ 1 200 000 zones déléguées. Faut-il 1 200 000 fils d'exécution ? Ou bien seulement 10 ? Ou entre les deux ? On peut toujours spéculer a priori sur le chiffre idéal. Il dépend de beaucoup de facteurs (matériel utilisé, efficacité de l'ordonnanceur utilisé, ici celui de Python, nombre de communications nécessaires avec les fils, etc). Notons que, le programme étant écrit en Python, où l'interpréteur lui-même n'est pas parallèle, une machine multiprocesseur n'est pas forcément utile.

Le mieux est de mesurer <https://www.bortzmeyer.org/mesurer-temps-execution.html> plutôt que d'essayer d'imaginer. Je fais un programme qui lit le fichier de zone de `.fr`, je crée N tâches Python et je confie à chacune 1 200 000 / N zones à interroger (chaque tâche doit évidemment aussi signaler à la tâche maîtresse le résultat de ses interrogations). L'interrogation se fait avec l'excellent module DNS Python <https://www.bortzmeyer.org/dnspython.html>, en ne parlant pas directement aux serveurs de noms mais en étant derrière un résolveur/cache Unbound (donc, attention, le cache peut influencer les mesures, c'est pour cela qu'elles ont été faites deux fois).

Avec seulement cent tâches (N = 100), le programme ne peut faire qu'environ cent zones par seconde. Je l'ai interrompu avant la fin. Avec 1000 tâches, on arrive à 480 zones par seconde. Et avec 10 000 tâches, on atteint 550 zones par seconde.

Cela illustre l'efficacité de l'ordonnanceur Python : il vaut mieux utiliser « beaucoup » de tâches.

Ne prenez pas le résultat au pied de la lettre pour **vos** programmes. Ce qui compte ici est la méthode (mesurer au lieu de supposer ; l'informatique est une science expérimentale). Le résultat, lui, dépend de beaucoup de facteurs.

Les tests ont été faits avec Python 2.5.2 sur une machine Debian `lenny` avec le noyau Linux 2.6.22. L'ordinateur était un Dell Poweredge avec huit gigaoctets de RAM et quatre processeurs AMD Opteron.