Présentation Perf

Jérémy ZANGLA

February 22, 2020

Contents

1 Présentation

1.1 Profiling

Le *profiling* est le processus qui analyse l'exécution d'un programme. Il permet d'étudier les performances de notre application, afin de trouver les points à améliorer. Les domaines que l'on peut étudier sont diverses : temps d'éxécution, espace mémoire, nombre et fréquence d'appel des fonctions. On appelle *profiler* l'outil qui nous permet de faire du profiling.

1.2 Outils

Il existe différents outils, appelés *profilers*, permettant le profiling en fonction du langage de programmation choisis. On citera par exemple :

- C/C++ : Valgrind (Callgrind, Cachegrind, ...), ...
- Java : Java Runtime Analysis Toolkit (JRAT), ...
- Python : cProfile, ...

Certains de ces outils doivent être intégrés au programme, d'autres sont lancés indépendamment en donnant le programme (les sources dans le cas d'un langage interprété).

2 Perf

2.1 Présentation

Le profiler traité ici est perf, il permet d'analyser des programmes développés dans plusieurs langages. Nous allons ici nous concentrer sur son utilisation sur des programmes en C++. C'est un programme qui crée un sous-système linux, permettant ainsi d'avoir son propre noyau pour acquérir les données d'exécution du programme analysé. Il est maintenu en parallèle du noyau linux et reçoit donc des mises à jour régulières. Il est séparé en plusieurs modules, chacun possédant des fonctions particulières. Cet utilitaire peut faire bien plus que ce qui sera présenté ici.

2.2 Installation

Ce logiciel n'est disponnible que sous Linux. La plupart des distributions permettent de l'installer directement depuis les dépots de leur gestionnaire de paquets. Cependant on peut l'installer manuellement depuis un linux, en clonant le dépôt git suivant : *git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git*. Il faut ensuite passer sur la branche correspondant au noyau du système utilisé. On peut ensuite se rendre dans le sous-répertoire *tools/perf*, et excécuter *make*. Si une erreur est levée, il faut vérifier que les programmes suivant sont installés : *make*, *flex* et *bisons*.

2.3 Configuration

2.3.1 Droits d'utilisation

Pour pouvoir excécuter perf, il faut par défaut avoir des droits super-utilisateurs. Il est cependant possible de changer ce comportement en changeant le contenu du fichier de configuration : */proc/sys/kernel/perf_event_paranoid*. Il faudra également modifié le fichier */etc/sysctl.conf* pour que les changements soient permanents. Le niveau par défaut est 3. Plus le niveau est faible, plus vous aurez accès à des parties du programme sans avoir besoin de droits supplémentaires (avec un minimum à -1 où l'accès est total). Sur une installation neuve, l'utilisation de *perf stat -a* vous indiquera les différents niveaux de privilèges possibles. La commande *perf test* permet de savoir quels évênements peuvent être analysés par perf dans la configuration actuelle.

2.3.2 Compilation

Vous pouvez utilisez perf sur tous les programmes. Cependant pour pouvoir avoir un lien avec le code source il faudra compilé en C++ avec les options -g et -pg. Sans ces options les noms de fonctions seront remplacés par leur adresse mémoire (sous forme hexadécimale).

2.4 Utilisation

2.4.1 Affichage des statistiques systèmes

```
1              perf stat executable
```