Profiling de programmes C et C++ avec GNU `gprof`

BERNARDI Fabrice, Université de Corse

version 0.5, 29/05/2003

Table des matières

1 Introduction

Le « profiling » d'un programme permet d'identifier les endroits où celui-ci passe le plus de temps, mais également quelles sont les fonctions qui sont exécutées et combien de fois. Un « profiler » est un programme capable d'analyser un exécutable. Le résultat de l'analyse est appelé un « profile ».

Tous les profilers fonctionnent sur le même schéma : ils collectent un certain nombre d'informations pendant l'exécution d'un programme. Ces informations sont dépendantes du contexte d'exécution et une comparaison ne pourra être effectuée entre deux profiles qu'à contexte égal. Une autre conséquence est qu'un profiler ne pourra être utilisé qu'après la fin de l'exécution du programme.

Le profiler standard dans le monde GNU est le programme `gprof`. Il fait partie du package `binutils` du projet GNU et est capable d'analyser des programmes écrits en C, C++, Pascal ou Fortran77.

Le profiling requiert trois étapes principales :

- Compilation et édition des liens avec des options spéciales permettant d'activer le profiling ;
- Exécution du programme avec génération automatique d'un fichier de données de profiling ;
- Exécution de `gprof` pour l'analyse du fichier de données.

Le profiling s'effectue par un changement dans la compilation de chacune des fonctions du programme. Les changements effectués permettent de compter le nombre de fois où une fonction est appelée ainsi que le temps passé à l'exécuter en suivant l'évolution d'un compteur. En général, ce compteur est relevé 100 fois par seconde

2 Compilation d'un programme pour le profiling

Le premier pas pour le profiling d'un programme est de le compiler en indiquant au compilateur d'intégrer les informations nécessaires. Dans le cas de GNU `gcc` (et donc des compilateurs `gcc`, `g++` ou `f77`), il faut rajouter l'option « `-pg` » en plus des options usuelles. Cette option permet d'altérer à la fois le comportement du compilateur et celui de l'éditeur de liens.

```
gcc monprog.c -o monprog -pg
```

Dans le cas d'un programme comportant plusieurs fonctions réparties dans plusieurs fichiers, on peut choisir de n'ajouter les informations de profiling qu'à certains d'entre elles. Cependant, les informations obtenues ne pourront plus être complètes et la seule disponible pour les fonctions qui n'auront pas été compilés avec l'option de profiling sera le temps total passé dans chacun d'entre elles.

3 Exécution du programme

Une fois le programme compilé pour le profiling, le programme doit être lancé de manière à générer les informations nécessaires. Ce lancement s'effectue de manière normale, avec la possibilité de passer des paramètres au programme.

Les profilers exigent que le programme se soient terminés normalement, soit par une terminaison normale de la fonction `main()`, soit par un appel à `exit(0)`. Un programme dont l'exécution a été interrompu (par un CTRL-C ou un faute de segmentation par exemple) ne pourront pas être analysés.

Dans le cas de GNU `gprof`, le fichier de données génér est appelé `gmon.out` et sera placé dans le répertoire courant dans lequel le programme a été lancé. Si ce fichier existe déjà dans le répertoire, il sera ecrasé.

Le profiler est appelé simplement de la manière suivante (en supposant que monprog est le programme à analyser) :

```
gprof monprog gmon.out
```

ou plus simplement par :

```
gprof monprog
```

3.1 Les options de la commande gprof

La forme complète de la commande gprof est la suivante :

```
gprof [options] [exécutable [fichier-de-données]] [> destination]
```

Si l'on omet le nom de l'exécutable, le fichier a.out est utilisé par défaut. Si l'on omet le nom du fichier de données, le fichier `gmon.out` est utilisé par défaut. Il est possible de fournir plusieurs fichiers de données, et dans ce cas, les différentes informations sont additionnées entre elles.

`gprof` accepte un certain nombre d'options, dont voici les plus importantes :

`-a` : Indique à gprof de supprimer l'affichage des fonctions déclarées privées. Leurs statistiques seront affectées à la fonction qui les appellent.

`-e nom_fonction` : Indique à `gprof` de ne pas afficher les informations sur la fonction `nom_fonction` et ses enfants.

`-E nom_fonction` : Indique à `gprof` de ne pas utiliser le temps d'exécution de la fonction dans le calcul des pourcentages.

`-f nom_fonction` : Indique à `gprof` de limiter l'affichage aux statitisques ne concernant que la fonction `nom_fonction`.

`-F nom_fonction` : Indique à `gprof` de n'effectuer ses statistiques que par rapport à la fonction `nom_fonction`.

`-b` : Indique à `gprof` de ne pas afficher les commentaires dans l'affichage généré.

Les résultats de l'analyse sont representés par deux tableaux : le « profil plat » et le « graphe d'appels ».

Le profil à plat (« flat profile ») montre combien de temps le programme passe dans chacune des fonctions et combien de fois chacune d'entre elle est appelée. Ce tableau permet donc de visualiser facilement quelles sont les fonctions qui consomment le plus de ressources.

Le graphe d'appels (« call graph ») montre, pour chacune des fonctions, quelles sont les fonctions qui l'appellent, quelles autres fonctions elle appelle et combien de fois elle le fait. Il présente également une estimation du temps passé dans chacune des sous-routines de chacune des fonctions.

3.2 Le graphe du profile à plat

Le profil à plat montre le temps total que le programme passe à exécuter chacune des fonctions. À moins que l'option `-z` ne soit spécifiée, les fonctions qui n'ont nécessité aucun temps de traitement ou n'ont été l'objet d'aucun appel ne sont pas mentionnées.

L'affichage d'un profil à plat est de la forme suivante :

```
Flat profile:

Each sample counts as 0.01 seconds.

 %   cumulative   self              self     total

time   seconds   seconds    calls  ms/call  ms/call  name

32.14      0.09     0.09  1279991     0.00     0.00  objet1::fonction1()

32.14      0.18     0.09    19999     0.00     0.01  objet1::fonction2()

21.43      0.24     0.06    20005     0.00     0.00  objet2::fonction2(char const*)

...
```

Les fonctions sont classées par ordre décroissant de temps de calcul. Deux fonctions spéciales, mcount et profil sont des fonctions de profiling et apparaissent dans chaque profil à plat. Leurs temps donnent une estimation de la surchage de calcul induite par le profiling.

Les différentes colonnes ont les significations suivantes :

% time : Pourcentage du temps d'exécution total passé à exécuter cette fonction.

cumulative seconds : Temps total cumulé que le processeur a passé à exécuter cette fonction, ajouté au temps passé à exécuter toutes les autres fonctions précédentes dans le tableau.

self seconds : Nombre de secondes effectivement passé à exécuter cette seule fonction.

calls : Nombre de fois où la fonction a été appelée. Si la fonction n'a jamais été appelée ou si son nombre d'appels n'a pas pu être déterminé, le champ est laissé vide.

self ms/calls : Estimation du nombre de millisecondes passées dans cette fonction par appel.

total ms/calls : Estimation du nombre de millisecondes passées dans cette fonction et dans ses enfants par appel.

name : Nom de la fonction.

3.3 Le graphe d'appels

Le graphe d'appels montre combien de temps est passé dans chacune des fonctions ainsi que dans chacun de ses enfants.

L'affichage d'un graphe d'appels est de la forme suivante :

```
                     Call graph (explanation follows)

granularity: each sample hit covers 4 byte(s) for 3.57% of 0.28 seconds

index % time    self  children    called     name

                                               <spontaneous>

[1]     96.0    0.00    0.27                 main [1]

                0.01    0.18       1/1           objet1::fonction1() [2]

                0.00    0.08       1/1           objet1::fonction2[param]() [6]

                0.00    0.00       1/1           objet1::fonction3() [16]

-----------------------------------------------

0.01    0.18       1/1           main [1]

[2]     68.4    0.01    0.18       1         objet1::fonction1() [2]

                0.00    0.16   20001/20001       objet2::fonction2(char *) [3]

                0.01    0.00   20000/39999       objet1::fonction3() [10]

                0.01    0.00   20001/20010       objet3::fonction4() [11]

...
```

Les lignes composées de tirets séparent le tableau en différentes entrées composées d'une ou plusieurs lignes. Dans chacune des entrées, la ligne primaire est celle qui débute par un nombre entre crochets. La fin de cette ligne spécifie la fonction concernée. Les lignes précédentes décrivent les fonctions appelantes de cette fonction et les lignes suivantes décrivent les fonctions appelées par cette fonction. Ces fonctions sont appelées les fonctions « enfants ». Toutes les entrées sont classées par temps passé dans la fonction et ses enfants.

3.3.1 La ligne primaire

La ligne primaire est la ligne qui désigne la fonction étudiée et donne les statistiques globales correspondantes.

Les différentes colonnes ont les significations suivantes :

index : Les entrées sont numérotées par des entiers consécutifs. Chaque fonction possède donc un index qui apparaît au début de la ligne primaire. Chacune des références à une fonction, en tant que fonction appelante ou fonction enfant, présente son index ainsi que son nom.

% time : Pourcentage du temps total passé dans la fonction, en incluant le temps passé dans chacune des fonctions enfants.

self : Temps total passé dans cette fonction.

children : Temps total passé dans les appels aux fonctions enfants.

called : Nombre total de fois où la fonction a été appelée. Si la fonction est appelée récursivement, il apparaît deux nombres séparés par un « + ». Le premier nombre désigne le nombre d'appels non-récursifs et le second le nombre d'appels récursifs.

name : Nom de la fonction. L'index est répété juste après ce nom. Si la fonction fait partie d'un cycle de récursion, le numéro du cycle est inscrit entre le nom de la fonction et son index.

3.3.2 Les lignes des fonctions parentes (fonctions appelantes de la fonction)

Une entrée de fonction présente une ligne pour chacune de ses fonctions parentes, c'est-à-dire pour chacune des fonctions qui l'ont appelée.

Les colonnes correspondent à celles présentes pour la ligne primaire, mais certaines de leurs significations ne sont pas les mêmes :

self : Estimation du temps passé dans la fonction enfant considérée lorsqu'elle a été appelée par sa fonction parente.

children : Estimation du temps passé dans chacune des fonctions enfants de la fonction enfant considérée lorsqu'elle a été appelée par sa fonction parente.

called : Le premier nombre indique le nombre de fois où la fonction enfant considérée a été appelée par sa fonction parente, tandis que le second nombre indique le nombre d'appels non-récursifs total.

name et index : Nom de la fonction parente et son index.

3.3.3 Les lignes des fonctions enfants

Une entrée de fonction présente une ligne pour chacune de ses fonctions enfants. De la même manière que précédemment, les colonnes correspondent à celles présentes pour la ligne primaire, mais certaines de leurs significations ne sont pas les mêmes :

self : Estimation du temps passé dans la fonction enfant considérée lorsqu'elle a été appelée par sa fonction parente.

children : Estimation du temps passé dans chacune des fonctions enfants de la fonction enfant considérée lorsqu'elle a été appelée par sa fonction parente.

called : Le premier nombre indique le nombre de fois où la fonction enfant considérée a été appelée par sa fonction parente, tandis que le second nombre indique le nombre d'appels non-récursifs total.

4 les visualisateurs

4.1 cgprof

cgprof est un visualisateur simple de fichier profile montrant de manière graphique les dépendances existantes entre les différentes focntions du programme. Il est composé d'un script shell téléchargeable à l'adresse suivante :

http://mvertes.free.fr/cgprof-1.0/cgprof

Il utilise le programme dotty inclus dans le package graphviz pour la visualisation :

```
gprof programme | cgprof -X11 > a.dot
dotty a.dot
```

4.2 kprof

kprof est un utilitaire du bureau KDE plus évolué que cgprof. Il est capable par exemple d'analyser les objets C++ de manière très ergonomique. Il est disponible à l'adresse :

http://kprof.sourceforge.net

5 Liens

http://www.gnu.org/manual/gprof-2.9.1/html_node/gprof_toc.html
http://www.cs.utah.edu/dept/old/texinfo/as/gprof.html