Empowered by Innovation NEC

Atelier
Profilage de codes de calcul

Premiers pas avec GPROF

Laurent Gatineau
Support applicatif
NEC HPC Europe

Ecole Centrale de Paris
11 juin 2014

Plan

- Présentation de l'outil GPROF
- Premiers pas avec GPROF
- Flat Profile
- Call Graph

Auteur: Laurent Gatineau

Présentation de l'outil GPROF

GPROF: GNU Profiler

- Support du C, C++ et Fortran.
- Compilateurs commerciaux supportent GPROF.
- Instrumentation du code et échantillonnage.

Quels types de profilage ?

- « Flat Profile »: Temps CPU par fonctions.
- « Call Graph »: Graphe d'appels des fonctions.

Interface: Mode ligne de commandes.
Outils externes:

- Xgprof (un peu ancien)
- KProf
- Gprof2dot

Auteur: Laurent Gatineau

Présentation de l'outil GPROF

Limites:

- Instrumentation du code:
  - Surcoût à l'exécution.
  - Le code de calcul et les librairies utilisés doivent avoir été compilés avec le support GPROF.
- Echantillonnage: Précision des temps de calcul limité.
- Pas d'information sur les librairies partagées.
- Comptabilise uniquement le temps CPU passé dans les fonctions: ne comptabilise pas les appels systèmes, les I/O…
- Pas de support OpenMP / Multi-thread.
- Le code de calcul doit se terminer « normalement ».

Auteur: Laurent Gatineau

Plan

- Présentation de l'outil GPROF
- Premiers pas avec GPROF
- Flat Profile
- Call Graph

Auteur: Laurent Gatineau

Premiers pas avec GPROF

- Compiler le code avec support GPROF:
  - Compilateurs GNU: flag '-pg'
  - Compilateurs Intel: flag '-p'
  - Compilateurs Open64: flag '-pg' / '-profile'
  - Compilateurs PGI: non supporté

```
gatineaul@service0:~/GPROF> make
gcc -g -pg -Wall   -c -o main.o main.c
gcc -g -pg -Wall   -c -o fun.o fun.c
gcc -g -pg -o test_gprof main.o fun.o
gatineaul@service0:~/GPROF>
```


Auteur: Laurent Gatineau

Premiers pas avec GPROF

- L'exécution du code se fait normalement:

```
gatineaul@service0:~/GPROF> ls
fun.c  fun.o  main.c  main.o  Makefile  test_gprof
gatineaul@service0:~/GPROF> ./test_gprof
Calling do_cpu...
Calling do_cpu_external...
Calling do_io...
Calling do_sleep...
gatineaul@service0:~/GPROF> ls
fun.c  fun.o  gmon.out  main.c  main.o  Makefile  test_gprof
gatineaul@service0:~/GPROF>
```

- Le fichier '`gmon.out`' contient les informations de profilage.

Premiers pas avec GPROF

- Choisir le nom du fichier de profilage est possible via la variable d'environnement `GMON_OUT_PREFIX`. Le PID du processus sera ajouté au préfixe:

```
gatineaul@service0:~/GPROF> export GMON_OUT_PREFIX=my_profile
gatineaul@service0:~/GPROF> rm gmon.out
gatineaul@service0:~/GPROF> ./test_gprof
Calling do_cpu...
Calling do_cpu_external...
Calling do_io...
Calling do_sleep...
gatineaul@service0:~/GPROF> ls
fun.c  fun.o  main.c  main.o  Makefile  my_profile.3242  test_gprof
gatineaul@service0:~/GPROF>
```


Auteur: Laurent Gatineau

Premiers pas avec GPROF

- La lecture du fichier de profilage se fait via la commande '`gprof`':

```
gatineaul@service0:~/GPROF> gprof -b -p test_gprof gmon.out
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 50.04     11.29    11.29        1    11.29    11.29  do_cpu_external
 49.96     22.56    11.27        1    11.27    11.27  do_cpu
  0.00     22.56     0.00        1     0.00     0.00  do_io
  0.00     22.56     0.00        1     0.00     0.00  do_sleep
gatineaul@service0:~/GPROF>
```


Auteur: Laurent Gatineau

Premiers pas avec GPROF

Quelques options à '`gprof`':

- -b / --brief : N'affiche pas les explications des différents champs.
- -p / --flat-profile: N'affiche que le profile à plat.
- -p[symbol] / --flat-profile=[symbol]: Filtre sur le symbole (peut-être répété).
- -q / --graph: N'affiche que le graphe d'appels.
- -q[symbol] / --graph=[symbol]: Filtre sur le symbole (peut-être répété).
- -s / --sum: Cumule ou assemble des profiles. Création d'un fichier '`gmon.sum`'.

Auteur: Laurent Gatineau

Flat Profile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 66.78     73.71    73.71     1444     0.05     0.05  csrMatrixVecProd
 20.07     95.86    22.15        1    22.15   110.29  cg
 13.07    110.29    14.43     2888     0.00     0.00  dotProd
  0.05    110.35     0.06        1     0.06     0.06  finiteDiffInitCsrMatrix
  0.01    110.36     0.01  1000000     0.00     0.00  f
  0.01    110.37     0.01        1     0.01     0.02  finiteDiffInitRhs
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixAllocate
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixFree
  0.00    110.37     0.00        1     0.00     0.00  meshInit
```

- `% time`: Temps total passé dans chaque fonction (% du temps d'exécution):
  - Identification rapide des « hot-spots ».

Auteur: Laurent Gatineau

Flat Profile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 66.73     73.71    73.71     1444     0.05     0.05  csrMatrixVecProd
 20.07     95.86    22.15        1    22.15   110.29  cg
 13.07    110.29    14.43     2888     0.00     0.00  dotProd
  0.06    110.35     0.06        1     0.06     0.06  finiteDiffInitCsrMatrix
  0.01    110.36     0.01  1000000     0.00     0.00  f
  0.01    110.37     0.01        1     0.01     0.02  finiteDiffInitRhs
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixAllocate
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixFree
  0.00    110.37     0.00        1     0.00     0.00  meshInit
```

- `cumulative seconds`: Temps cumulé passé dans les fonctions (en secondes).

Flat Profile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 66.78     73.71    73.71     1444     0.05     0.05  csrMatrixVecProd
 20.07     95.86    22.15        1    22.15   110.29  cg
 13.07    110.29    14.43     2888     0.00     0.00  dotProd
  0.05    110.35     0.06        1     0.06     0.06  finiteDiffInitCsrMatrix
  0.01    110.36     0.01  1000000     0.00     0.00  f
  0.01    110.37     0.01        1     0.01     0.02  finiteDiffInitRhs
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixAllocate
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixFree
  0.00    110.37     0.00        1     0.00     0.00  meshInit
```

- `self seconds`: Temps passé dans chaque fonction (en secondes). N'inclus pas les sous-fonctions.

Auteur: Laurent Gatineau

Flat Profile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 66.78     73.71    73.71     1444     0.05     0.05  csrMatrixVecProd
 20.07     95.86    22.15        1    22.15   110.29  cg
 13.07    110.29    14.43     2888     0.00     0.00  dotProd
  0.05    110.35     0.06        1     0.06     0.06  finiteDiffInitCsrMatrix
  0.01    110.36     0.01  1000000     0.00     0.00  f
  0.01    110.37     0.01        1     0.01     0.02  finiteDiffInitRhs
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixAllocate
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixFree
  0.00    110.37     0.00        1     0.00     0.00  meshInit
```

- `calls`: Nombre de fois où chaque fonction est appelée.

Auteur: Laurent Gatineau

Flat Profile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
 66.78     73.71    73.71     1444    0.05     0.05  csrMatrixVecProd
 20.07     95.86    22.15        1   22.15   110.29  cg
 13.07    110.29    14.43     2888    0.00     0.00  dotProd
  0.05    110.35     0.06        1    0.06     0.06  finiteDiffInitCsrMatrix
  0.01    110.36     0.01  1000000    0.00     0.00  f
  0.01    110.37     0.01        1    0.01     0.02  finiteDiffInitRhs
  0.00    110.37     0.00        1    0.00     0.00  csrMatrixAllocate
  0.00    110.37     0.00        1    0.00     0.00  csrMatrixFree
  0.00    110.37     0.00        1    0.00     0.00  meshInit
```

- `self s/call`: Temps passé dans chaque fonction par appel (en secondes). N'inclus pas les sous-fonctions. Attention: c'est une moyenne !

Auteur: Laurent Gatineau

Flat Profile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 66.78     73.71    73.71     1444     0.05     0.05  csrMatrixVecProd
 20.07     95.86    22.15        1    22.15   110.29  cg
 13.07    110.29    14.43     2888     0.00     0.00  dotProd
  0.05    110.35     0.06        1     0.06     0.06  finiteDiffInitCsrMatrix
  0.01    110.36     0.01  1000000     0.00     0.00  f
  0.01    110.37     0.01        1     0.01     0.02  finiteDiffInitRhs
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixAllocate
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixFree
  0.00    110.37     0.00        1     0.00     0.00  meshInit
```

- `self s/call`: Temps passé dans chaque fonction par appel (en secondes). Inclus les sous-fonctions (si connues de gprof).

Auteur: Laurent Gatineau

Flat Profile

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 66.78     73.71    73.71     1444     0.05     0.05  csrMatrixVecProd
 20.07     95.86    22.15        1    22.15   110.29  cg
 13.07    110.29    14.43     2888     0.00     0.00  dotProd
  0.05    110.35     0.06        1     0.06     0.06  finiteDiffInitCsrMatrix
  0.01    110.36     0.01  1000000     0.00     0.00  f
  0.01    110.37     0.01        1     0.01     0.02  finiteDiffInitRhs
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixAllocate
  0.00    110.37     0.00        1     0.00     0.00  csrMatrixFree
  0.00    110.37     0.00        1     0.00     0.00  meshInit
```

- Attention à la précision de l'échantillonnage...

Auteur: Laurent Gatineau

Flat Profile – HPL librairies statiques

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 79.86     25.74    25.74                             mkl_blas_avx_dgemm_kernel_0
  6.35     27.79     2.05 200020000     0.00     0.00  HPL_rand
  4.37     29.20     1.41 200883228     0.00     0.00  HPL_lmul
  2.17     29.90     0.70 200861700     0.00     0.00  HPL_ladd
  1.58     30.41     0.51       40     0.01     0.01  HPL_dlaswp00N
  1.49     30.89     0.48 200840168     0.00     0.00  HPL_setran
  0.99     31.21     0.32                             mkl_blas_avx_dtrsm_ker_llu_a4_b8
  0.68     31.43     0.22        2     0.11     2.44  HPL_pdmatgen
  0.47     31.58     0.15        5     0.03     0.03  HPL_pdlange
  0.40     31.71     0.13                             mkl_blas_avx_dgemm_copybn
  0.37     31.83     0.12                             mkl_blas_avx_dgemm_kernel_1_0
```

- Information parcellaire si l'option de compilation '`-pg`' n'a pas été utilisée correctement…

Auteur: Laurent Gatineau

Flat Profile – HPL librairies partagées

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 38.25      2.54     2.54 200020000     0.00     0.00  HPL_rand
 24.78      4.19     1.65 200883228     0.00     0.00  HPL_lmul
 14.61      5.16     0.97 200861700     0.00     0.00  HPL_ladd
  8.13      5.70     0.54       40     0.01     0.01  HPL_dlaswp00N
  6.93      6.16     0.46 200840168     0.00     0.00  HPL_setran
  3.61      6.40     0.24        2     0.12     2.95  HPL_pdmatgen
  2.26      6.55     0.15        5     0.03     0.03  HPL_pdlange
  0.68      6.59     0.05                             HPL_equil
  0.60      6.63     0.04   820162     0.00     0.00  HPL_jumpit
```

- Aucune information sur les librairies partagées…
- Non géré par `gprof`… Alternative: `sprof`.

Auteur: Laurent Gatineau

Plan

- Présentation de l'outil GPROF
- Premiers pas avec GPROF
- Flat Profile
- Call Graph

Call Graph

Call Graph (Graphe d'appels des fonctions):

- Connaître l'enchainement des appels de fonctions.
- Suivi des paramètres lors des appels de fonctions.
- Identification des fonctions qui ne sont jamais appelées.
- Savoir par qui une fonction est appelée souvent.
- Attention aux fonctions récursives.

Call Graph

```
gatineaul@service0:~/GPROF/CallGraph> gprof -b -q test_gprof gmon.out
                            Call graph
granularity: each sample hit covers 4 byte(s) for 0.04% of 25.72 seconds
index % time    self  children    called     name
                0.10    0.00       1/256         fun2 [4]
               25.62    0.00     255/256         fun1 [3]
[1]    100.0   25.72    0.00     256         do_cpu [1]
-----------------------------------------------
                                                <spontaneous>
[2]    100.0    0.00   25.72                 main [2]
                0.00   25.62       1/1           fun1 [3]
                0.00    0.10       1/1           fun2 [4]
-----------------------------------------------
                0.00   25.62       1/1           main [2]
[3]     99.6    0.00   25.62       1         fun1 [3]
               25.62    0.00     255/256         do_cpu [1]
-----------------------------------------------
                0.00    0.10       1/1           main [2]
[4]      0.4    0.00    0.10       1         fun2 [4]
                0.10    0.00       1/256         do_cpu [1]
-----------------------------------------------
Index by function name
   [1] do_cpu (main.c)         [3] fun1 (main.c)          [4] fun2 (main.c)
```


Auteur: Laurent Gatineau

Call Graph

```
granularity: each sample hit covers 4 byte(s) for 0.04% of 25.72 seconds
index % time    self  children    called     name
                0.10    0.00       1/256         fun2 [4]
               25.62    0.00     255/256         fun1 [3]
[1]    100.0   25.72    0.00     256         do_cpu [1]
-----------------------------------------------
                                                <spontaneous>
[2]    100.0    0.00   25.72                 main [2]
                0.00   25.62       1/1           fun1 [3]
                0.00    0.10       1/1           fun2 [4]
-----------------------------------------------
                0.00   25.62       1/1           main [2]
[3]     99.6    0.00   25.62       1         fun1 [3]
               25.62    0.00     255/256         do_cpu [1]
-----------------------------------------------
                0.00    0.10       1/1           main [2]
[4]      0.4    0.00    0.10       1         fun2 [4]
                0.10    0.00       1/256         do_cpu [1]
```

- `%time`: Pourcentage du temps total passé dans la fonction et les sous-fonctions.

Auteur: Laurent Gatineau

Call Graph

```
index % time    self  children    called     name
                0.10    0.00       1/256          fun2 [4]
               25.62    0.00     255/256          fun1 [3]
[1]    100.0   25.72    0.00     256            do_cpu [1]
-----------------------------------------------
                0.00   25.62       1/1            main [2]
[3]     99.6   0.00   25.62       1            fun1 [3]
               25.62    0.00     255/256          do_cpu [1]
-----------------------------------------------
                0.00    0.10       1/1            main [2]
[4]      0.4   0.00    0.10       1            fun2 [4]
                0.10    0.00       1/256          do_cpu [1]
```

- `self`: Temps passé dans la fonction. Non inclus les sous fonctions.
- Remarque: décomposition en fonction des appelants.

Auteur: Laurent Gatineau

Call Graph

```
index % time    self  children    called     name
                0.10    0.00       1/256          fun2 [4]
               25.62    0.00     255/256          fun1 [3]
[1]    100.0   25.72    0.00     256            do_cpu [1]
-----------------------------------------------
                                                <spontaneous>
[2]    100.0    0.00   25.72                   main [2]
                0.00   25.62       1/1            fun1 [3]
                0.00    0.10       1/1            fun2 [4]
-----------------------------------------------
```

- `children`: Temps passé dans les sous-fonctions.
- Remarque: décomposition en fonction des appelés.

Auteur: Laurent Gatineau

Call Graph

```
index % time    self  children    called     name
                0.10    0.00       1/256         fun2 [4]
               25.62    0.00     255/256         fun1 [3]
[1]    100.0   25.72    0.00     256           do_cpu [1]
-----------------------------------------------
                                                <spontaneous>
[2]    100.0    0.00   25.72                   main [2]
                0.00   25.62       1/1           fun1 [3]
                0.00    0.10       1/1           fun2 [4]
-----------------------------------------------
```

- `called`: Nombre de fois où la fonction est appelée.
- Remarque: décomposition en fonction des appelant.

Auteur: Laurent Gatineau

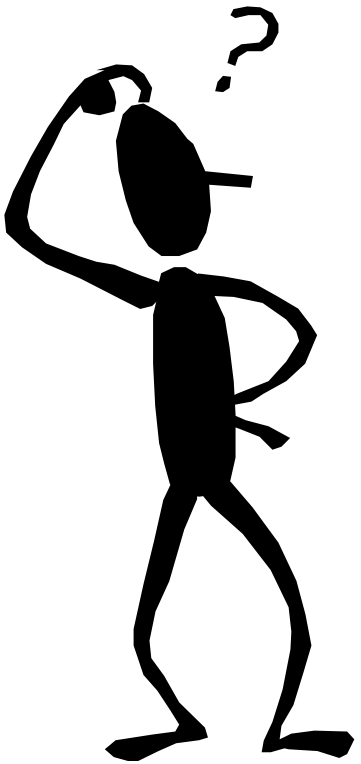
Call Graph

```
index % time    self  children    called     name
                0.10    0.00       1/256       fun2 [4]
               25.62    0.00     255/256       fun1 [3]
[1]    100.0   25.72    0.00     256         do_cpu [1]
-----------------------------------------------
                                               <spontaneous>
[2]    100.0    0.00   25.72                 main [2]
                0.00   25.62       1/1         fun1 [3]
                0.00    0.10       1/1         fun2 [4]
-----------------------------------------------
                0.00   25.62       1/1         main [2]
[3]     99.6    0.00   25.62       1         fun1 [3]
               25.62    0.00     255/256       do_cpu [1]
-----------------------------------------------
```

- `name`: Nom des fonctions + index.
- Remarques:
  - Décalage entre la fonction et les fonctions appelantes / appelées.
  - Au-dessus: fonctions appelantes.
  - En-dessous: fonctions appelées.

Auteur: Laurent Gatineau

QUESTIONS ?



Auteur: Laurent Gatineau

Travaux Pratiques

- Récupérez le programme de test dans `/home/gatineaul/TP_gprof/main.c`
- Compilez le programme pour pouvoir faire un profilage avec GPROF.
- Exécutez le programme en mesurant le temps de restitution et le temps utilisateur.

Travaux Pratiques

- Affichez le profile de l'application.
- Le profile donne un temps cumulé en seconde, à quoi correspond-t-il ?
- Identifiez le ou les hot spots.
- Identifiez la ou les fonctions les plus appelées.

Auteur: Laurent Gatineau

Travaux Pratiques

- Affichez le graphe d'appel.
- Quelles sont les fonctions qui appellent la fonction la plus consommatrice en temps ?
- Y-a-t-il une relation entre le nombre de fois où la fonction est appelée et le temps attribué à chaque appelant ?
  - Jetez un œil aux sources…
- GPROF peut annoter les sources avec l'option '`-A`'. Que voyez-vous ?
- Essayez avec votre code de calcul préféré…