GPROF

Introdução à Avaliação de Desempenho e Dependabilidade de Sistemas

Professor: Paulo Maciel – prmm@cin.ufpe.br
Alunos: Ronierison Maciel – rsm4@cin.ufpr.br
Paulo Pereira – prps@cin.ufpe.br

Agenda

- O que é o Gprof?
- Por que usar um *profiler?*
- Como utilizar o Gprof?
  - Compilando um programa para *profiling*
  - Executando um programa para gerar *profile data*
  - Executando o Gprof
- Interpretando a saída do Gprof
  - Flat profile
  - Call graph

O que é o Gprof?

- GProf é uma ferramenta para análise dinâmica da execução de programas (*profiler*).

- Ele produz um “perfil” da execução de programas C, Pascal ou Fortran77.

- Este tipo de informação permite analisar a quantidade de métodos (ou funções) existentes no código.

- Essa ferramenta pode ser usada em conjunto com o GCC.

O que é o Gprof?

Gprof reporta informação de tempo de execução de cadauma das funções do programa.

Gprof também mostra uma árvore de chamada de funções do programa, ou seja, podemos verificar o tempo de execução de uma dada função e de suas sub-rotinas.

Por que usar um *profiler*?

- *Profiling* é importante porque nos permite verificar ONDE o programa está gastando tempo.

- Essa informação mostra quais partes do programa estão executando mais lentamente do que você esperava.

- Essas partes do programa (funções) são candidatas a serem reescritas visando uma melhora no desempenho do seu programa.

Por que usar um *profiler*?

Como o Gprof utiliza uma informação gerada em tempo de execução do programa, então ele pode ser usado em programas que são relativamente extenso e complexo para serem analisados simplesmente lendo o código fonte.

Como utilizar o Gprof?

São 5 passos:

1. Tenha um programa que funcione (O Gprof não é um *debugger*);
2. Compilar e linkar o programa com a opção de *profiling* habiititada “-pg”;
3. Executar o programa normalmente;
4. Executar o Gprof e
5. Analisar a informação gerada pelo Gprof.

Como utilizar o Gprof?

Caso o compilador e o Gprof não estiverem instalado, utilizem os comandos:

```
user@user:~$ sudo apt install build-essential
user@user:~$ sudo apt install binutils
```

Como utilizar o Gprof?

Essencialmente, o código-fonte precisa ser compilado com a opção de profiling habilitada;

Para habilitar está opção, adicione as flags **-pg** ao processo de compilação;

Vejamos por exemplo para compilar um código-fonte em C utilizando o GCC (**G**nu **C**ompiler **C**ollection).

```
user@user:~$ gcc -Wall -pg codigo_fonte.c -o programa
```

Como utilizar o Gprof?

O Gprof irá compilar seu código-fonte e criar um programa executável capaz de produzir as informações necessárias para sua análise com o Gprof;

Após a compilação, seu programa pode ser executado, pois somente após a execução será possível obter informações sobre o tempo consumido pelos métodos e rotinas do seu programa;

Como utilizar o Gprof?

Como se observa, seu programa deve ser executado naturalmente como de constume e produzir as saídas normalmente esperadas, além de criar um arquivo chamado **gmon.out** caso termine sem erros.

Note que seu programa deve ter permissão para escrever no diretório onde o mesmo está rodando, caso contrário será lançado um erro.

Como utilizar o Gprof?

Executando o Gprof

Agora que temos nosso *profile data* “gmon.out”, podemos executar o Gprof.

Exemplo:

user@user:~$ **gprof** programa **gmon.out** > analysis.txt

Interpretando a saída do Gprof

Ao analisar a saída do Gprof, no nosso exemplo o arquivo chamado de “analysis.txt”, este arquivo pode ser aberto em qualquer editor de texto. Percebemos que ele é dividido em duas seções:

1)Flat profile e

2)Call graph.

Interpretando a saída do Gprof

Flat profile: Mostra a percentagem de tempo que um programa com alguns métodos de ordenação de vetores gasta em cada um dos seus métodos.

%time – Indica a percentagem de tempo de execução gasto por método;

cumulative seconds – É o tempo gasto por um método acrescido do tempo gasto pelos métodos anteriores na tabela;

self seconds – Mostra o tempo gasto em segundos por método;

calls – Mostra o número de vezes que um método foi chamado;

self s/ calls – Representa o tempo em segundos gasto em um método para cada chamada;

total s/ calls – Mostra o tempo gasto em segundos por um método e suas sub-rotinas e

name – Identificação do nome do método ou rotina.

Interpretando a saída do Gprof

Call graph: Exibe a quantidade de tempo gasto em cada método e suas sub-rotinas, na forma de um grafo direcionado e acíclico.

index – Representa um ídice para cada método ou rotina;

%time – indica a percentagem de tempo gasto pelo método e por suas sub rotinas;

self – Representa o tempo gasto pelo método;

children – Representa o tempo gasto pelo “filhos” de um método;

called – Indica o número de vezes que este método foi chamado e

name – Representa o nome do método analisado.

Gprof - opções

Existem diversas opções de execução do Gprof

Algumas opções interessantes são:

Gprof - opções

GPROF(1) GNU GPROF(1)

NAME top

gprof - display call graph profile data

SYNOPSIS top

```
gprof [ -[abcDhilLrsTvwxyz] ] [ -[ACeEfFJnNOpPqQZ][name] ]
 [ -I dirs ] [ -d[num] ] [ -k from/to ]
 [ -m min-count ] [ -R map_file ] [ -t table-length ]
 [ --[no-]annotated-source[=name] ]
 [ --[no-]exec-counts[=name] ]
 [ --[no-]flat-profile[=name] ] [ --[no-]graph[=name] ]
 [ --[no-]time=name] [ --all-lines ] [ --brief ]
 [ --debug[=level] ] [ --function-ordering ]
 [ --file-ordering map_file ] [ --directory-path=dirs ]
 [ --display-unused-functions ] [ --file-format=name ]
 [ --file-info ] [ --help ] [ --line ] [ --inline-file-names ]
 [ --min-count=n ] [ --no-static ] [ --print-path ]
 [ --separate-files ] [ --static-call-graph ] [ --sum ]
 [ --table-length=len ] [ --traditional ] [ --version ]
 [ --width=n ] [ --ignore-non-functions ]
 [ --demangle[=STYLE] ] [ --no-demangle ]
 [--external-symbol-table=name]
 [ image-file ] [ profile-file ... ]
```

Gprof - opções

A opção --function-ordening

Faz com o que o Gprof mostre uma sugestão de "ordenação" de chamada de funções do programa baseado no *profile data*.

Essa sugestão tem a intenção de melhorar paginação, tlb e comportamento cache (para programas em sistemas que suportem ordenação arbitrária de funções em um executável)

tlb é uma cache que o hardware de gerenciamento de memória utiliza para melhorar a velocidade da traduação de endereços virtuais.

Gprof - opções

A opção -a

Se existem funções estáticas no programa e você deseja omiti-las do *profile*, utilize a opção **-a** e

Se você não deseja mais que o Gprof imprima arquivo de saída as definições de cada coluna, você pode suprimir essa informações extras com a flag **-b**.

Gerando um DOT Graph

Existe uma ferramenta que recebe como entrada a informação geraa pelo Gprof (e também outros profilers) e converte essa informação em um DOT Graph.

O ferramental chama-se “Gprof2Dot”.

```
user@user:~$ apt install python graphviz
```

https://github.com/jrfonseca/gprof2dot

Exercício

Desenvolvam um programa em C que contenha funções e cada uma das delas execute uma tarefa. Avaliem o desempenho da mesma com utilizando o Gprof e em seguida plotem com Gprof2Dot.