Universidade do Vale do Itajaí
Campus Kobrasol

Análise de Algoritmos

Avaliação M2

Aluno: Maurício Macário de Farias Junior

Professor: Antonio Carlos Sobieransk

São José, 2018, 16 de Fevereiro

Resumo

Neste trabalho será apresentado duas ferramentas de profilling que foram instaladas e testadas, será descrito as especificações relevantes e suas características, o que a ferramenta lhe permite fazer, e que tipos de gráficos mostra.

Desenvolvimento

Aqui será apresentado a ferramenta Valgrind e a ferramenta gprof

Valgrind

Valgrind é uma ferramenta usada para memory debugging, detecção de memory leak e profilling, é na essência uma máquina virtual usando técnicas de compilação just-in-time(JIT), incluindo recompilação dinâmica, Valgrind traduz o programa em uma forma mais simples e temporária chamada Representação Intermediata, que é processador-neutro.

O programa roda muito mais lento, chegando a um quinto da velocidade do programa normal.

Abaixo está a saida da ferramenta do Valgrind usada para checagem de memory leak, e logo abaixo o código executado pela ferramenta

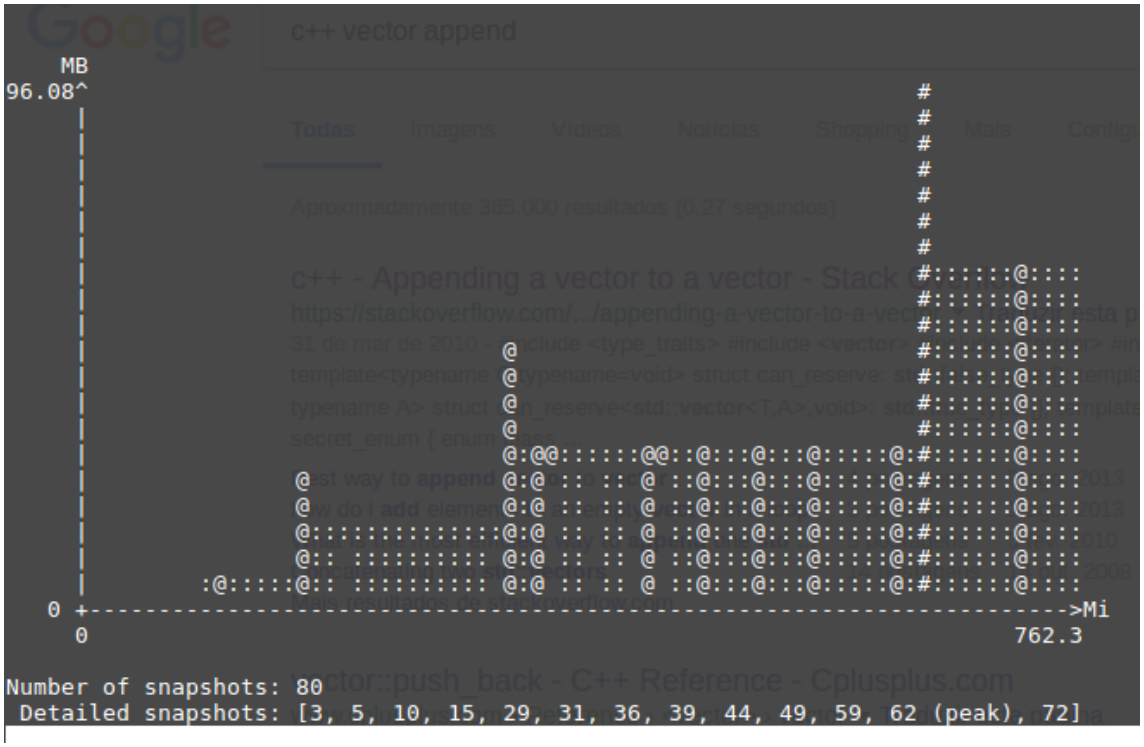
```
draetus@mauricio-PC ~/Desktop/test $ valgrind ./fail
==2806== Memcheck, a memory error detector
==2806== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==2806== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==2806== Command: ./fail
==2806==
==2806==
==2806== HEAP SUMMARY:
==2806==     in use at exit: 122,704 bytes in 20,001 blocks
==2806==   total heap usage: 20,001 allocs, 0 frees, 122,704 bytes allocated
==2806==
==2806== LEAK SUMMARY:
==2806==    definitely lost: 50,000 bytes in 20,000 blocks
==2806==    indirectly lost: 0 bytes in 0 blocks
==2806==      possibly lost: 0 bytes in 0 blocks
==2806==    still reachable: 72,704 bytes in 1 blocks
==2806==         suppressed: 0 bytes in 0 blocks
==2806== Rerun with --leak-check=full to see details of leaked memory
==2806==
==2806== For counts of detected and suppressed errors, rerun with: -v
==2806== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
void fail()
{
        int* x = new int(10);
        char *k = new char('x');
}

int main()
{
        for (int i=0; i<10000; i++)
        {
                fail();
        }
        return 0;
}
```

Nota-se que a criação dos ponteiros e seu preenchimento mas a falta de seu liberamento gera memory leak que é detectado pelo Valgrind

Abaixo está apresentado a saida da ferramenta massif do Valgrind, e logo abaixo o código executado para tal saida.

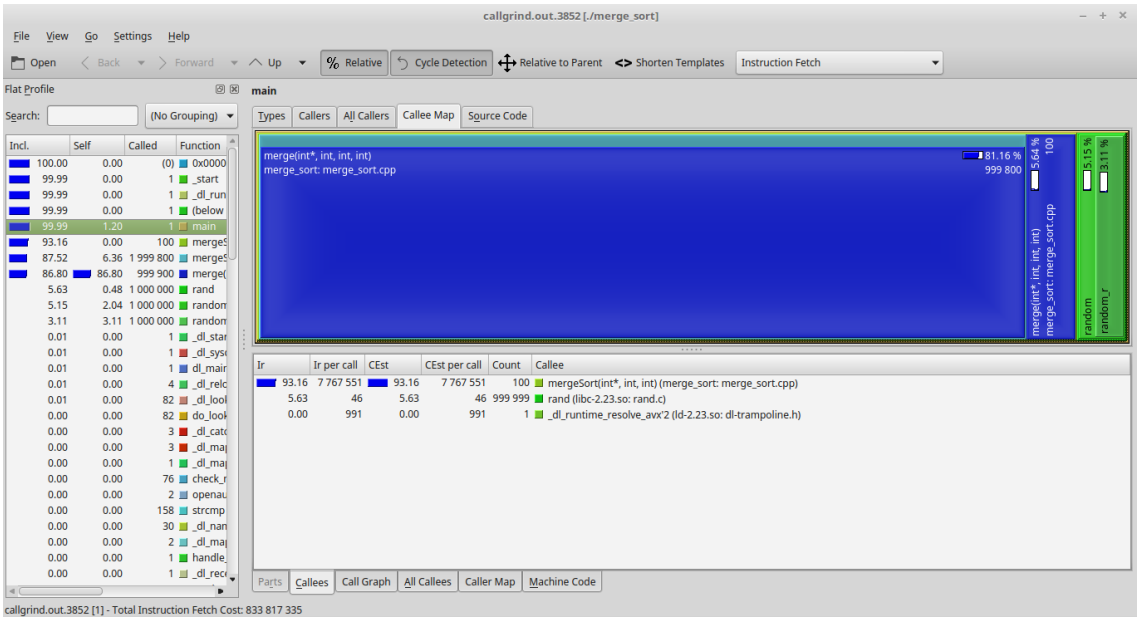



```cpp
#include <vector>

int main()
{
        int i;
        std::vector<int> arr;
        for (i=0; i<10000000; i++)
        {
                arr.push_back(i);
        }
        return 0;
}
```

Pode-se perceber que o crescimento do vetor dinâmico gera um aumento no uso de memória que é detectado pela ferramenta.

Abaixo está apresentado o gráfico gerado que mostra a porcentagem de tempo gasto em cada função do código, gerado pela ferramenta Callgrind do Valgrind e exibido pelo kcachegrind

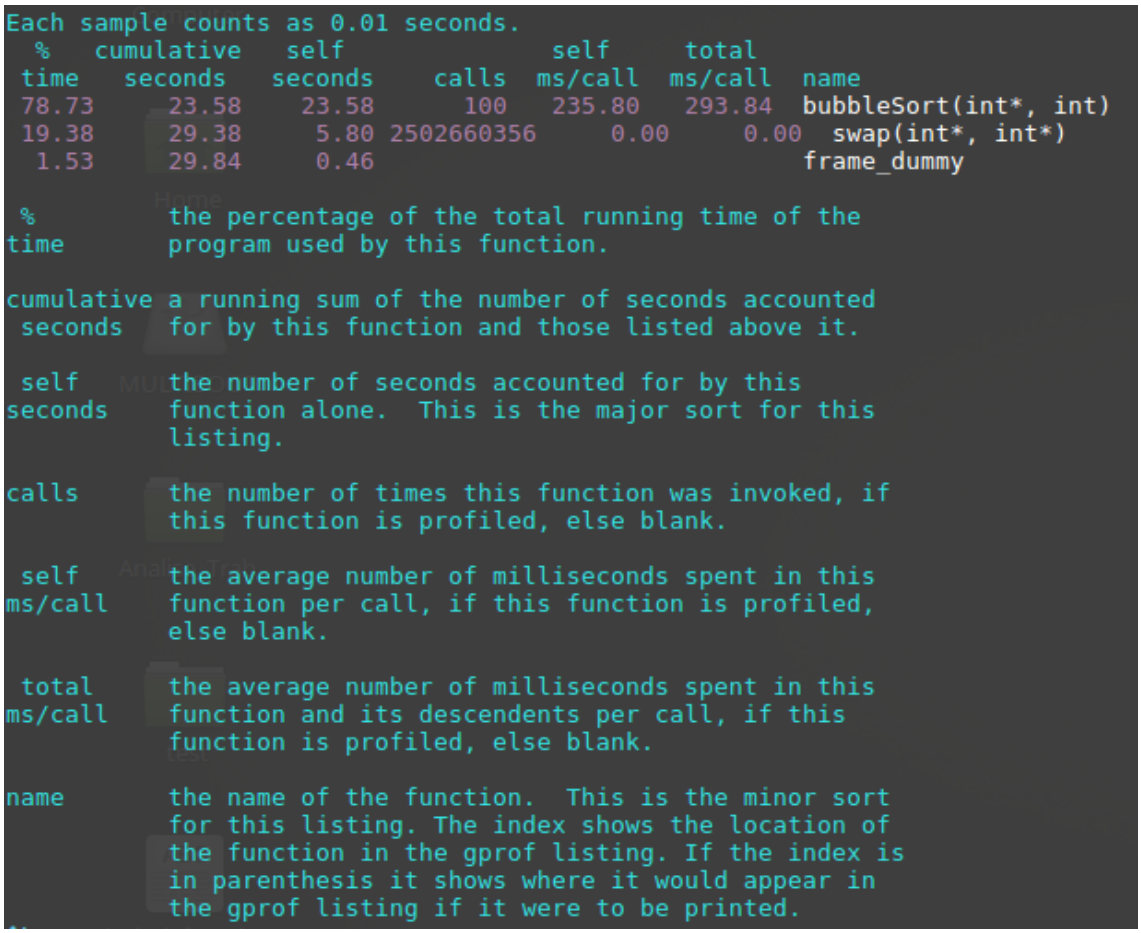


O algoritmo Merge Sort nesse teste está ordenando um vetor de 10000 posições preenchido aleatóriamente 100 vezes.

Gprof

Gprof é uma ferramenta de análise de perfomance e é capaz de gerar pequenos gráficos, não é totalmente exato mas é estatisticamente aproximado.

Abaixo está apresentado o tempo de execução do algoritmo de ordenação Bubble Sort



```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 78.73     23.58    23.58      100   235.80   293.84  bubbleSort(int*, int)
 19.38     29.38     5.80 2502660356     0.00     0.00  swap(int*, int*)
  1.53     29.84     0.46                             frame_dummy

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.
```

O algoritmo está ordenando um vetor de 10000 posições preenchido aleatóriamente 100 vezes.

Conclusão

Conclui-se que as ferramentas de profilling tem um propósito muito importante e conseguem exibir dados muito úteis com muita eficiência e precisão, com várias opções de exibições de dados, assim como gráficos ou a simples impressão de dados.