



Memória Cache

Este trabalho tem como objetivo demonstrar como as operações de acesso à memória são relevantes no desempenho geral de um algoritmo. Além disso, também está no escopo o aprendizado sobre como utilizar ferramentas de análise de desempenho.

1. Apresente a organização da cache do computador que será utilizado para realizar os experimentos, como por exemplo seu tamanho, associatividade, etc. Utilize o site: <http://www.cpu-world.com>;
2. Pesquise quatro algoritmos de ordenação: *Bubblesort*, *Radixsort*, *Quicksort* e mais um a sua escolha. Descreva **brevemente** como é o funcionamento de cada um;
3. Execute cada um dos algoritmos e meça o desempenho deles utilizando a ferramenta *perf* (use pelo menos as *tags* cache-references, cache-misses, task-clock, cycles e instructions). A linguagem dos algoritmos deve ser C ou C++. Observação: os algoritmos podem ser retirados de sites e afins, entretanto devem ser **devidamente referenciados**. A Figura 1 exemplifica a saída desse comando;

```
Performance counter stats for './exec':

0,416605      task-clock (msec)      #    0,508 CPUs utilized
894.852      cycles                  #    2,148 GHz
613.401      instructions            #    0,69  insn per cycle
31.278       cache-references        #   75,078 M/sec
10.860       cache-misses            #   34,721 % of all cache refs

0,000820227 seconds time elapsed
```

Figura 1. Saída do comando *perf*.

4. Relate os dados obtidos e compare os resultados de cada um dos algoritmos para entradas diferentes (uma pequena, por exemplo 1000 números, e uma grande, 10000 números);
5. Realize o mesmo procedimento, porém agora utilizando a ferramenta *valgrind*. Como ela é um simulador de memória cache, é possível alterar os tamanhos dos níveis, associatividade, etc. Faça alguns testes alterando esses parâmetros e relate as mudanças obtidas. A Figura 2 exemplifica a saída desse comando. Observação: neste tópico, execute apenas uma vez e com entrada pequena, visto que isso é uma simulação, ou seja, a execução para entradas grandes seria muito demorada;

```

[SAÍDA DO SEU CÓDIGO]
==15912==
==15912== I   refs:      152,908
==15912== I1 misses:      892
==15912== LLi misses:      887
==15912== I1 miss rate:    0.58%
==15912== LLi miss rate:    0.58%
==15912==
==15912== D   refs:      50,463 (38,918 rd + 11,545 wr)
==15912== D1 misses:      2,926 ( 2,363 rd +   563 wr)
==15912== LLd misses:      2,443 ( 1,933 rd +   510 wr)
==15912== D1 miss rate:    5.8% (  6.1% +   4.9% )
==15912== LLd miss rate:    4.8% (  5.0% +   4.4% )
==15912==
==15912== LL refs:      3,818 ( 3,255 rd +   563 wr)
==15912== LL misses:      3,330 ( 2,820 rd +   510 wr)
==15912== LL miss rate:    1.6% (  1.5% +   4.4% )

```

Figura 2. Saída do comando *valgrind*.

- Escolha **qualquer algoritmo** e modifique-o para aproveitar a memória cache e/ou seus princípios, de forma que o seu desempenho seja alterado. Explique o que foi feito e compare antes e depois da modificação. Observação: nessa tarefa, não podem existir algoritmos iguais, ou seja, **o de cada grupo deve ser único**. Desse modo, um documento será compartilhado com a turma contendo os algoritmos que já foram escolhidos. Envie um email para o professor (ou monitor, caso tenha) com a sua escolha.

Forma de entrega: PVANet, em um único arquivo compactado.

O que deve ser entregue:

- Documentação do trabalho;
- Arquivos fonte que foram modificados.