Avaliação de desempenho na execução de aplicações

Alberto José Proença & João Garcia Barbosa

Temática

Os exercícios propostos têm como objectivos:

- a aquisição de **técnicas de medida de tempos de execução** (microscópicos) **de programas elaborados em C**, e respectiva compreensão/ interpretação de resultados, ao nível do *assembly*;
- a realização de **medições de desempenho das variantes de optimização dum dado programa** (evoluções para programas mais eficientes), e posterior análise de algumas das técnicas de optimização utilizadas.

Contexto

Técnicas de medição

Uma das melhores formas de exprimir o **desempenho da execução de um programa** que efectua essencialmente um cálculo repetitivo - normalmente sob a forma de um ciclo – é utilizando uma métrica que relacione o tempo necessário para realizar o conjunto de operações que se efectua sobre cada um dos elementos da lista (repetitiva), com uma característica temporal do CPU, tornando a métrica independente da frequência do *clock* do CPU. Assim, a métrica que iremos utilizar é baseada na proposta no texto de apoio em 5.2: o número médio de **ciclos de *clock* por elemento** processado **(CPE)**.

Uma **forma simplificada para se obter o valor de CPE** (mas pouco precisa), consiste em **(i)** inicializar um "cronómetro", **(ii)** executar o programa de teste (a medir), **(iii)** contabilizar o número de ciclos de *clock* desde a última inicialização do contador e **(iv)** calcular o respectivo valor por elemento, dividindo o nº total de ciclos pelo nº de elementos da lista.

Para efectuar **medições de tempos "microscópicos" com grande precisão**, alguns processadores das últimas gerações disponibilizam facilidades extras com este objectivo. Contudo, a precisão dos resultados pode ser influenciada por diversos factores do sistema de computação, conforme referido em 9.4.1 e 9.4.2:

No caso concreto do IA32 (apenas após o Pentium), **a Intel disponibiliza um contador de 64 bits** (o *cycle counter*) **e uma nova instrução,** **`rdtsc`** (de "*read time stamp counter*"). Os programas em C que permitem o manuseamento deste contador como "cronómetro" podem ser analisados em `clock.h` e `clock.c`. Mais detalhes na secção 9.3 do texto de apoio.

Uma análise deste código (também disponível na Fig. 9.9 do texto de apoio) mostra a **inserção de código em *assembly* directamente no ficheiro com o programa em C** (característica não normalizada de acordo com ANSI C). A inserção de código *assembly* em linha é tratado no texto de apoio em 3.15, onde em 3.15.2 se mostra, de maneira simples, como se faz a utilização do comando `asm`, reconhecido pelo `gcc`, com uma sintaxe extremamente simples:

```
asm( code-string [ : output-list [ : input-list [ : overwrite-list ]]]);
```

Exercícios

1. Desenvolva um programa em C que determina a frequência de relógio do sistema de computação onde executar.
   Sugestão: implemente uma rotina em C para aceder ao contador de ciclos de relógio do processador, utilizando a instrução *assembly* RTDSC, e depois utilize essa rotina em conjunto com uma função do Linux.

2. A multiplicação de matrizes é uma das operações de álgebra linear frequentemente usada para cálculo científico. O algoritmo básico para multiplicar duas matrizes quadradas NxN é:

```
For(i=0; i < N; i++)
    For(j=0; j < N; j++)
        For(k=0; k < N; k++)
            C(i,j) += A(i,k) x B(k,j)
```

   a) Implemente uma rotina em C que permita multiplicar duas matrizes quadradas e calcule os valores de CPE para as dimensões N = 2^x (x = 4…16).

      (i) No cálculo do CPE o nº total de elementos a considerar é o nº de resultados a calcular, i.e., a dimensão total da matriz de resultados C (neste caso, NxN).

      (ii) Neste exercício considere que os elementos das matrizes são valores reais de precisão simples.

      (iii) Na selecção dos valores de tempos medidos a confiar, use a abordagem `k_best`, com `k=3`, conforme descrita no livro de Bryant & O'Hallaron (apontadores na página Web dos sumários).

      (iv) Para a inicialização das matrizes A, B e C em memória, use (obrigatoriamente) a função `init_sq_arrays (arg1, arg2, arg3)`, sendo os 3 primeiros argumentos respectivamente os apontadores para as matrizes A, B e C, e o último a dimensão N da matriz; esta função inicializará em memória (apenas), as matrizes A e B com valores aleatórios entre 0 e 1, e a matriz C com zeros. O ficheiro objecto com esta função encontra-se na Web (link na página Web dos sumários).

   b) Analise e proponha estratégias para melhorar o desempenho deste algoritmo. Mostre e comente alguns dos resultados obtidos.