Avaliação de desempenho na execução de aplicações

Trabalho para Casa: **TPC8**

Alberto José Proença

Prazos

Entrega **impreterível**: semana de 02-Dez-02, na hora e local da sessão Teórico-Prática.

Não serão aceites trabalhos entregues depois deste prazo.

Metodologia

Idêntica aos trabalhos anteriores.

Introdução

A lista de exercícios que se apresenta segue directamente o material apresentado nas aulas teóricas da semana 10 (ver sumários na página da disciplina na Web), requerendo os conceitos básicos adquiridos em aulas anteriores e os ficheiros com algum código C já disponibilizado (ver *link* nos sumários).

Contexto

Transformação sistemática de um programa noutro mais eficiente

Os exercícios propostos têm como objectivo 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*; o 2º exercício é a preparação para o próximo trabalho, o qual irá efectuar **medições de desempenho das variantes de optimização dum dado programa** (evoluções para programas mais eficientes), para posterior análise das técnicas de optimização utilizadas.

O programa a estudar é o **exemplo de optimização** analisado no texto de apoio (de Bryant) na secção 5.3: parte de uma simples estrutura de um vector de dados – *Vector Abstract Data Type*, adiante designado por vector ADT – para **combinar todos os elementos dum vector num único valor**, de acordo com uma dada operação. O conjunto de programas em C relevantes para este trabalho, apresentados no livro nos capítulos 5 e 7, são disponibilizados em formato electrónico, de modo compactado (zip), junto a este enunciado (respectivamente nas pastas `opt` e `perf`).

O vector ADT permite a definição de vários tipos de dados como elementos constituintes do vector; neste exercício, iremos apenas considerar **os tipos `int` e `float`**. Os ficheiros relacionados com o vector ADT são o `vec.h` e o `vec.c`. Mais detalhes no texto de apoio (em 5.3).

O exemplo de optimização – que combina os valores dum vector – encontra-se nos ficheiros `combine.h` e `combine.c`. Usando diferentes definições das constantes `IDENT` e `OPER`, o código pode ser recompilado para efectuar operações distintas aplicadas aos dados. Iremos apenas considerar **as operações de soma e multiplicação**. Mais detalhes no texto de apoio (em 5.3).

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 (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:

- **o nº de processos que simultaneamente partilham o CPU** durante a execução do programa em estudo; **para reduzir/eliminar a influência deste factor**, este caso de estudo utiliza um programa com um tempo de execução pequeno (< 7 mseg), garantindo assim uma muito baixa probabilidade de interrupção (i.e, baixa probabilidade de ocorrência de um *context switching*);
- **os diferentes tempos de acesso aos diversos níveis da hierarquia de memória; para minimizar este factor**, este caso de estudo foi dimensionado para caber todo na *cache* mais rápida (1º nível), e são disponibilizadas **rotinas para "aquecer" a *cache*** antes da medição dos tempos de execução (disponíveis em `time_p.c`)

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. Escreva um programa em C que lhe permita estimar a frequência do *clock* do processador que está a usar, usando a função `sleep` com o argumento 1 (unidade: segundo), e as funções que lidam com o *cycle counter*. Execute esse programa num computador à escolha.

Guarde o programa numa disquete, e traga-o para a aula teórico-prática para repetir o ensaio.

Nota: se tiver dificuldade com o programa, analise o código disponibilizado em `clock.c`.

2. Escreva um (ou mais) programa(s) em C para calcular o CPE (de forma simplificada) da função `combine1` (se quiser, adapte-os do ficheiro `combine.c`), quando aplicado a um vector com 1000 elementos, para as seguintes operações: soma de inteiros, soma de reais, produto de inteiros, produto de reais (por "reais", entenda-se do tipo `float`).

Faça a compilação sem qualquer optimização (use a opção –g). Teste num computador à sua escolha, e leve o programa para a aula para fazer medições num computador do laboratório.