Medição
do desempenho
Computacional

Medição do Tempo de Execução de um Programa

Embora para dada combinação de programa ou dados, a máquina execute um número fixo de instruções

E a execução das instruções seja controlada pelo relógio do processador, que é regulado por um oscilador de precisão

Há muitos factores que podem variar de uma execução de um mesmo programa para outra, porque os computadores não executam um programa de cada vez mas sim em simultâneo, alternando entre processos

Medição do Tempo de Execução de um Programa

Factores que influenciam a calendarização exacta dos recursos do processador para um programa:

- Número de utilizadores a partilharem o sistema
- Tráfego na rede
- Tempo das operações no disco
- O acesso às caches dependem não só das referências feitas pelo programa que tentamos medir, mas também dos processos concorrentes
- A lógica de “branch prediction” é afectada pela história anterior, que pode variar de uma execução do programa para a outra.

Medição do Tempo de Execução de um Programa

As variações nos tempos devidas às mudanças de contexto tendem a ser muito grandes e logo devem ser eliminadas.

As variações provocadas por outros factores, tais como cache e branch prediction, são normalmente resolvidas avaliando a operação do programa sob condições cuidadosamente controladas.

Normalmente usam-se máquinas especialmente configuradas para minimizar fontes de irregularidades na medição de desempenhos. Por exemplo, limitam-se os acessos e desactivam-se muitos serviços do sistema operativo e de rede.

Mas, não há uma resposta definitiva sobre como medir o tempo de execução de um programa, para um programa arbitrário a correr num sistema arbitrário, porque há demasiadas variações de mecanismos de temporização, comportamentos dos sistemas operativos e ambiente de execução.

Pipelining

Imagine-se um microprocessador simples em que todas as instruções têm dois passos: descodificação e execução

O microprocessador poupa tempo se descodificar uma instrução enquanto a anterior está a ser executada.

A este princípio chama-se **pipelining**.

Em microprocessadores avançados o pipeline pode ter vários níveis, de modo a que várias instruções consecutivas possam ser processadas ao mesmo tempo, uma em cada nível do pipeline.

Branch prediction (Previsão de Derivações)

Uma instrução de derivação (branch) é a implementação de uma construção if-then-else.

Se a condição for verdadeira, então salta para outra localização, senão continua com a instrução seguinte.

Isto provoca uma quebra na sequência de instruções do pipeline, porque o processador não sabe qual é a instrução que vem a seguir até executar a instrução de derivação.

Quanto maior for o pipeline, mais tempo se vai ter de esperar até o processador saber qual instrução introduzir de seguida no pipeline.

Branch prediction (Previsão de Derivações)

Branch Prediction é a solução para este problema.

Consiste numa tentativa do processador prever se a instrução de derivação vai saltar ou não, baseada nos registos do que esta derivação realizou anteriormente.

O processador decide sobre qual instrução carregar no pipeline baseando-se nesta previsão, antes de ter a certeza. A isto chama-se execução especulativa.

Se a previsão se verificar errada, o processador tem de limpar o pipeline e desconsiderar todos os cálculos baseados nesta previsão, mas se estiver certa, poupou-se imenso tempo.

Medição do Tempo de Execução de um Programa

Existem dois mecanismos básicos usados pelos computadores para medir a passagem do tempo:

• um é baseado num temporizador de baixa frequência, que interrompe periodicamente o processador. Este temporizador pode ser acedido através de chamadas a funções das bibliotecas.

• outro é baseado num contador que é incrementado a cada ciclo de relógio

Medição do Tempo de Execução de um Programa

Na medição do desempenho computacional é necessário ainda ter em atenção que a programação feita numa linguagem de alto nível poderá não ter a mesma semântica da linguagem máquina produzida pelo compilador e interpretada pelo processador.

Para além de que, como já foi referido, a análise e interpretação dessas mesmas medidas estarem condicionadas a múltiplos parâmetros, variáveis, e arquitecturas inerentes à maquina a testar.

Daí que o conhecimento completo dessas arquitecturas é ponto importante para a correcta avaliação e justificação dos resultados das medidas efectuadas.

Medição do Tempo de Execução de um Programa

Vamos estudar duas abordagens possíveis, para a implementação dos meios de medidas temporais:

- o **registo RDTSC** da arquitectura IA32
- e a função **gettimeofday()**

A estratégia base consiste na utilização das referidas primitivas de forma a contabilizar temporalmente o tempo dispendido na execução de um determinado código.

RDTSC – Read Time Stamp Counter

- É uma instrução
  - para testes de referência (benchmarking)
  - de dois bytes: 0F 31
  - que retorna uma contagem de 64 bits nos registos EDX:EAX
    - os 32 bits de ordem mais elevada no EDX
    - e os 32 bits de ordem mais baixa no EAX
  - com o número de ciclos de relógio desde que o CPU foi activado ou reactivado

- Atenção que a medida é dada em **ciclos**, não em tempo.

- # segundos = # ciclos / frequência

Registos

O registo é a forma de armazenamento mais básica e mais rápida que existe para o programador.

Os registos residem no próprio processador e são a única forma de armazenamento existente em todo o computador à qual o processador pode aceder directamente.

Os programas usam registos como espaço de armazenamento temporário, para os dados que estejam a usar no momento.

Ter registos de 32 bits permite ao processador trabalhar com 32 bits de cada vez.

Um grupo de 32 bits é chamado uma "**doubleword**", de 16 bits é chamado de "**word**" e 8 de bits de **byte**.

Função gettimeofday()

É uma função que permite obter a data e a hora

Obtém a hora actual, expressa em segundos e microsegundos, desde 1 de Janeiro de 1970, 00:00 (UTC), e armazena-a em *tp*.

tp é um apontador para uma estrutura *timeval* que guarda a hora actual.

Se for bem sucedida, gettimeofday() retorna 0,
senão retorna -1 e, neste caso, é armazenado um código de erro na variável global *errno*

O código de erro que gettimeofday() pode colocar em errno é:
[EFAULT], indica que *tp* é um apontador inválido

Função gettimeofday()

Sinopse da função:

```
#include<sys/time.h>
int gettimeofday(struct timeval *tp, void *tpz);
```

tzp não é usado e deve ser null.

A estrutura timeval está definida em <sys/time.h>, como:

```
struct timeval {
        time_t  tv_sec;  /* segundos desde 1 de Janeiro de 1970 */
        long  tv_usec;  /* e microsegundos */
};
```