

Arquitetura e Organização de Computadores II

Desempenho

Prof. Nilton Luiz Queiroz Jr.

Desempenho

- O que significa dizer que um computador é mais rápido que o outro?



Desempenho

- Um computador X é mais rápido que um computador Y quando o tempo de resposta em X é menor que o tempo de resposta para Y na mesma aplicação;
- Assim, quando dizemos que X é N vezes mais rápido que Y, isso significa:

$$\frac{\text{Execution time}_Y}{\text{Execution time}_X} = n$$



Desempenho

- Como:

$$\text{Execution time} = \frac{1}{\text{Performance}}$$

- Temos:

$$n = \frac{\text{Execution time}_Y}{\text{Execution time}_X} = \frac{\frac{1}{\text{Performance}_Y}}{\frac{1}{\text{Performance}_X}} = \frac{\text{Performance}_X}{\text{Performance}_Y}$$

Desempenho

- Quando se diz que o throughput de X é 1,3 vezes maior que o de Y, significa que o número de tarefas completadas por unidade de tempo do computador X é 1,3 vezes o número completado por Y;



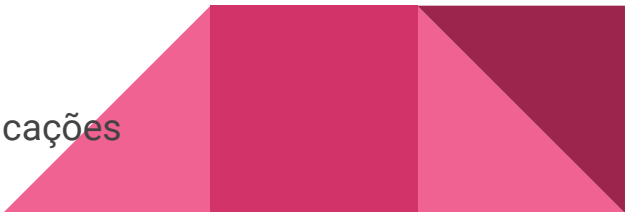
Desempenho

- Existem diversas maneiras de medir o desempenho, porém a mais consistente e confiável é o tempo;
- Até mesmo o tempo de execução pode ser definido de diferentes maneiras;
 - Tempo de “relógio”:
 - O tempo que o usuário final vê com resultado;
 - Tempo de CPU:
 - O tempo que a aplicação em questão gastou na CPU para executar;



Desempenho

- Para medir o desempenho é comum o uso de benchmarks;
- A melhor escolha de benchmarks para medir o desempenho é o uso de aplicações reais;
- Executar programas que são mais simples que aplicações reais induzem a erros;
 - Alguns exemplos desse tipo de aplicação:
 - *Kernels*;
 - Rotinas de aplicações grandes;
 - *Toy Programs*;
 - Programas com poucas linhas;
 - *Benchmarks Sintéticos*;
 - Programas que tentam ter comportamento de aplicações reais



Benchmarks

- Usam-se pacotes de benchmarks (*benchmarks* suites) para avaliar desempenho dos processadores;
 - Um exemplo desses pacotes de *benchmarks* é o desenvolvido pela SPEC (Standard Performance Evaluation Corporation);
 - Versões que iniciaram na década de 80 e evoluíram até hoje;
- A vantagem de pacotes é que o ponto fraco de um *benchmark* que compõe o pacote é contornado pela presença dos outros;



Princípios no projeto de computadores

- No projeto e na análise de computadores é importante levar em conta alguns princípios:
 - Tirar vantagens do paralelismo;
 - Princípio da localidade;
 - Foco no caso comum;



Princípios no projeto de computadores

- Tirar vantagem do paralelismo:
 - Paralelismo com vários discos e processadores;
 - Típico para servidores;
 - Paralelismo em nível de instrução;
 - Paralelismo em nível de projetos
 - Por exemplo:
 - Caches associadas por conjunto utilizam vários bancos de memória que normalmente são pesquisados em paralelo;
 - ULAs atuais utilizam carry-lookahead;



Princípios no projeto de computadores

- Princípio de localidade:
 - Programas tendem a reutilizar dados e instruções;
 - Dois tipos de localidade são os observados:
 - Temporal
 - Itens acessados recentemente serão acessados num futuro próximo;
 - Espacial;
 - Itens com endereços próximo costumam ser referenciados em curto espaço de tempo;



Princípios no projeto de computadores

- Foco no caso comum
 - Ao fazer uma escolha favoreça o caso que ocorre com mais frequência;
 - Por exemplo: Unidades de busca e decodificação são usadas com mais frequência que um multiplicador, portanto é preferível otimizá-las primeiro;
 - Para quantificar esse princípio aplica-se uma lei fundamental, a lei de Amdahl;



Lei de Amdahl

- A Lei de Amdahl é usada para calcular o ganho de desempenho de alguma parte do computador;
 - Estabelece que o ganho de desempenho alcançado melhorando algum modo mais rápido é limitado pela proporção de tempo que esse modo mais rápido pode ser usado;
 - Ela define o ganho de velocidade que pode ser obtido usando um recurso particular;
 - O ganho de velocidade é expresso pela seguinte razão:

$$\text{Speedup} = \frac{\text{Performance for entire task using the enhancement when possible}}{\text{Performance for entire task without using the enhancement}}$$

ou ainda

$$\text{Speedup} = \frac{\text{Execution time for entire task without using the enhancement}}{\text{Execution time for entire task using the enhancement when possible}}$$

Lei de Amdahl

- A lei de Amdahl concede um modo rápido de obter o ganho de velocidade dada uma melhoria, e depende de dois fatores:
 - Fração do tempo de computação no computador que pode ser convertida para tirar proveito da melhoria;
 - A melhoria obtida pelo modo de execução melhorado;



Lei de Amdahl

- A fração do tempo de computação que pode ser convertida para tirar proveito da melhoria é chamada de **fração melhorada**
 - Sempre menor o igual a 1;
 - Por exemplo: um programa que leva 60 segundos, e desse tempo 20 segundos podem ser melhorados, logo $20/60$ será a fração melhorada;
- A melhoria obtida pelo modo de execução melhorado significa o quão rápido a tarefa seria executada se o modo melhorado fosse usado para o programa todo, e é chamado de ganho de velocidade (speedup) melhorado;
 - Ou seja, o quão mais rápido o modo melhorado é que o modo original;
 - Esse valor é sempre maior que 1;
 - Por exemplo: modo melhorado leva 2 segundos e o modo original leva 5 segundos, o ganho de velocidade melhorado é de $5/2$,

Lei de Amdahl

- O tempo de execução usando o computador original com o modo melhorado será o tempo gasto usando a parte não melhorada somado ao tempo gasto usando a melhoria;

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

- O ganho de velocidade geral é a razão dos tempos de execução

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Exemplo

- Uma transformação comum exigida nos processadores de gráficos é a raiz quadrada. Suponha que a raiz quadrada em ponto flutuante seja responsável por 20% do tempo de execução de um benchmark gráfico. Uma alternativa é incrementar o desempenho somente da operação de raiz quadrada em um fator de 10. Outra alternativa é que todas operações de ponto flutuante tenham seu fator incrementado em 1,6. Ambas teriam o mesmo esforço para serem implementadas. Sabe-se que 50% do tempo desse benchmark é gasto com as operações de ponto flutuante. Compare as duas alternativas.



Exemplo

- Melhorar a Raiz
 - Fração melhorada: 0,2;
 - Ganho de velocidade melhorado: 10;



Exemplo

- Melhorar a Raiz
 - Fração melhorada: 0,2;
 - Ganho de velocidade melhorado: 10;

$$\frac{1}{(1 - 0.2) + \frac{0.2}{10}} = \frac{1}{0.82} = 1.22$$



Exemplo

- Melhorar todas operações em FP;
 - Fração melhorada: 0,5;
 - Ganho de de velocidade melhorado: 1,6;



Exemplo

- Melhorar todas operações em FP;
 - Fração melhorada: 0,5;
 - Ganho de de velocidade melhorado: 1,6;

$$\frac{1}{(1 - 0.5) + \frac{0.5}{1.6}} = \frac{1}{0.8125} = 1.23$$



Princípios no projeto de computadores

- Essencialmente todos os computadores são construídos usando um clock que trabalha em uma taxa constante;
 - Esses eventos de tempo são chamados de ticks, ticks de clock, ciclos ou ciclos de clock;
- O período de um desses eventos é referenciado de acordo com a duração ou frequência;
 - Unidades de medidas adotadas podem ser:
 - Segundos;
 - 1ns por exemplo;
 - Hertz:
 - 1GHz por exemplo;

Obs: $\text{Hz} = 1/\text{s}$ ou s^{-1}

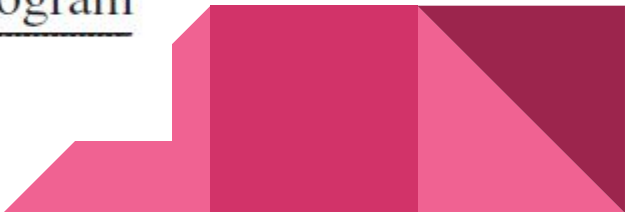


Princípios no projeto de computadores

- O tempo de CPU de um programa pode ser expresso de duas maneiras:

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

ou

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$


Princípios no projeto de computadores

- Além do número de ciclos de clock pode-se também medir o número de instruções executadas;
- Se soubermos o número de ciclos de clock e o total de instruções pode-se também calcular o número de ciclos de clock por instrução;
 - Clock cycles per instruction - CPI;

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

- Outra alternativa ao CPI é o número de instruções por clock;
 - Instructions per Clock - IPC
- IPC é o inverso do CPI;

Princípios no projeto de computadores

- O tempo de CPU pode ser dado pela seguinte equação:

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

- Note que o tempo de ciclo pode ser obtido pelo inverso da frequência;
- Sabendo que:

$$\text{Instruction count} = \frac{\text{Instructions}}{\text{Program}}$$

$$\text{Cycles per instruction} = \frac{\text{Clock cycles}}{\text{Instruction}}$$

$$\text{Clock cycle time} = \frac{\text{Seconds}}{\text{Clock cycle}}$$

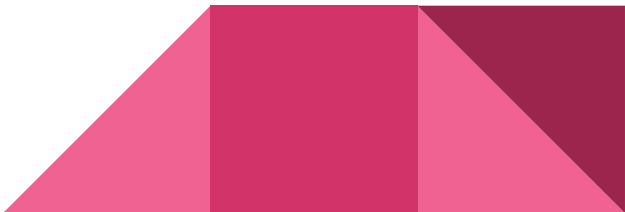


Princípios no projeto de computadores

- Temos:

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}}$$

- Assim

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}}$$


Princípios no projeto de computadores

- Isso mostra que a performance de um computador depende de principais coisas:
 - Tempo do ciclo do clock;
 - Depende da tecnologia e organização do hardware
 - CPI;
 - Organização do hardware e conjunto de instruções;
 - Total de instruções;
 - Conjunto de instruções e compilador;



Princípios no projeto de computadores

- Uma outra maneira de calcular a quantidade o total de ciclos pela seguinte fórmula:

$$\text{CPU clock cycles} = \sum_{i=1}^n \text{IC}_i \times \text{CPI}_i$$

- Sendo IC_i a quantidade de instruções do tipo i
- CPI_i a quantidade média de clocks por instrução de instruções do tipo i ;
 - Note que uma mesma instrução pode ter CPI diferentes, pois existem diferentes eventos que alteram o tempo que a instrução levou para ficar pronta, isso faz como CPI_i deva ser medido, e não obtido da tabela;
 - Hazard no pipeline;
 - Miss em cache;
 - Etc;

Princípios no projeto de computadores

- Usando essa fórmula temos o tempo de CPU dado por:

$$\text{CPU time} = \left(\sum_{i=1}^n \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

- O CPI geral pode ser calculado como:

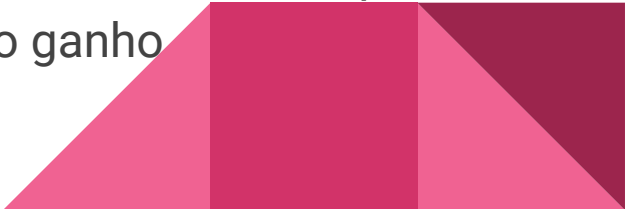
$$\text{CPI} = \frac{\sum_{i=1}^n \text{IC}_i \times \text{CPI}_i}{\text{Instruction count}} = \sum_{i=1}^n \frac{\text{IC}_i}{\text{Instruction count}} \times \text{CPI}_i$$

- Desse modo se usa o CPI médio de cada instrução e a fração de ocorrência de cada instrução no programa;

Exemplo

- Considere novamente o problema do cálculo de raiz quadrada, porém com um novo benchmark onde:
 - A frequência de operações com ponto flutuante é de 25%;
 - O CPI dessas operações é de 4;
 - As demais instruções, (que não se enquadram nas operações de ponto flutuante) tem CPI de 1,33;
 - As frequência de operações de raiz quadrada é de 2%;
 - O CPI das operações de raiz é 20;

O que é melhor: Reduzir o CPI das operações de ponto flutuante de 4 para 2,5 ou reduzir o CPI das operações de raiz para 2? Qual o ganho de velocidade alcançado por cada uma das duas abordagens?



Exemplo

Inicialmente Calculamos o CPI atual:

$$CPI_{Total}(Atual) = Frequência(FP) \times CPI(FP) + Frequência(ExcetoFP) \times CPI(ExcetoFP)$$

$$CPI_{Total}(Atual) = 0,25 \times 4 + 0,75 \times 1,33 = 2$$

Assim temos que o CPI atual é de 2. Com isso podemos calcular o CPI das outras operações exceto as de raiz quadrada

$$CPI_{Total}(Atual) = Frequência(Raiz) \times CPI(Raiz) + Frequência(ExcetoRaiz) \times CPI(ExcetoRaiz)$$

$$CPI(ExcetoRaiz) = \frac{CPI_{Total}(Atual) - Frequência(Raiz) \times CPI(Raiz)}{Frequência(ExcetoRaiz)}$$

$$CPI(ExcetoRaiz) = \frac{2 - 0,02 \times 20}{98} = 1,633$$

Exemplo

- Agora que temos todas as variáveis podemos calcular as duas alternativas;
 - Cálculo para operações FP com CPI = 2,5:

$$CPI_{Total}(FP_{2,5}) = Frequência(FP_{2,5}) \times CPI(FP_{2,5}) + Frequência(ExcetoFP) \times CPI(ExcetoFP)$$

$$CPI_{Total}(FP_{2,5}) = 0,25 \times 2,5 + 0,75 \times 1,33 = 1,625$$

$$Speedup(FP_{2,5}) = \frac{2}{1,625} = 1,23$$

Observe que as frequências não mudam e o CPI das operações que não são ponto flutuante também não;



Exemplo

- Cálculo para operações de raiz com CPI = 2:

$$CPITotal(Raiz_2) = Frequência(Raiz_2) \times CPI(Raiz_2) + Frequência(ExcetoRaiz) \times CPI(ExcetoRaiz)$$

$$CPITotal(Raiz_2) = 0,02 \times 2 + 0,98 \times 1,633 = 1,64$$

$$Speedup(Raiz_2) = \frac{2}{1,64} = 1,22$$



Exercícios

1. Um hardware A com pipeline com execução fora de ordem e despacho múltiplo executou uma quantidade de 2.000.000 instruções num total de 2.500.000 ciclos de clock. Um outro B hardware com um pipeline similar, porém com especulação, executou as mesmas 2.000.000 instruções em 1.500.000 ciclos de clock.
 - a. Qual o CPI de cada um deles?
 - b. Qual o speedup alcançado com a especulação para esse caso?
 - c. Se a frequência do clock é de 2 GHz, qual o tempo de CPU para cada um deles?
2. Suponha que estejamos considerando melhorar uma máquina adicionando a ela hardware vetorial. Quando um processamento é executado em modo vetor nesse hardware, é 10 vezes mais rápido do que o modo original de execução. Que porcentagem de vetorização é necessária para atingir um ganho de velocidade de 2?