

Desempenho

A rápida taxa de melhoria na tecnologia de computadores veio em decorrência de dois fatores: avanços na tecnologia utilizada na construção de computadores e inovação no projeto de computadores.

O projeto de computadores implica em determinar que atributos são importantes para a nova máquina e, então, projetar a máquina a fim de maximizar o desempenho, mantendo-se dentro dos limites de custo.

Os aspectos mais importantes a observar são o projeto do conjunto de instruções, a organização funcional, o projeto lógico e a implementação.

Na otimização do projeto, as métricas mais importantes são *custo* e *desempenho*.

Desempenho

Tempo de resposta (tr) ou *tempo de execução* (te) corresponde ao tempo entre o começo e o fim de um evento.

Dizer que uma máquina A é n vezes mais rápida que uma máquina B significa que:

$$\frac{\text{tempo de execução}_B}{\text{tempo de execução}_A} = n$$

Desempenho é definido como o inverso do tempo de execução:

$$n = \frac{\text{tempo de execução}_B}{\text{tempo de execução}_A} = \frac{\frac{1}{\text{desempenho}_B}}{\frac{1}{\text{desempenho}_A}} = \frac{\text{desempenho}_A}{\text{desempenho}_B}$$

Desempenho

O tempo de execução inclui acessos a disco, acessos à memória, entrada e saída, execuções do sistema operacional.

Tempo de CPU (tc) significa o tempo que a CPU consome computando, não incluindo o tempo de espera para entrada e saída ou para executar outros programas.

Tempo de CPU do usuário (tcu) é o tempo de CPU gasto no programa.

Tempo de CPU do sistema (tcs) é o tempo de CPU gasto com o sistema operacional para realizar tarefas requeridas pelo programa.

Ex: $tcu=90.7s$, $tcs=12.9s$, $tr=159s \Rightarrow tc$ corresponde a 65% de tr

Desempenho

Desempenho do sistema é utilizado para se referir ao tempo de resposta em um sistema não carregado.

Desempenho da CPU é utilizado para se referir ao tempo de CPU do usuário em um sistema não carregado.

Throughput corresponde ao total de trabalho realizado em um dado tempo.

O computador que realiza a mesma quantidade de trabalho no menor tempo é o mais rápido – tempo de execução (uma tarefa) ou *throughput* (várias tarefas).

Desempenho

As decisões de projeto devem favorecer os casos mais freqüentes: tornar rápido o caso mais comum.

A lei de Amdahl define o *speedup* (S), que consiste do ganho em desempenho que pode ser obtido ao melhorar determinada característica do computador:

$$S = \frac{\text{desempenho de toda a operação usando a melhoria}}{\text{desempenho de toda a operação sem usar a melhoria}}$$

$$S = \frac{\text{tempo de execução de toda a operação sem usar a melhoria}}{\text{tempo de execução de toda a operação usando a melhoria}}$$

Desempenho

O *speed up* (S), a partir de alguma melhoria, depende de dois fatores:

1. A fração do tempo de computação na máquina original que pode tirar vantagem da melhoria ($F_{melhoria}$): se 20s do tempo de execução de um programa, que leva 60s para ser executado, podem ser melhorados, a fração é 20/60;
2. O ganho obtido com a execução da melhoria ($S_{melhoria}$): se a melhoria leva 2s para ser executada e a original leva 5s, o ganho é 5/2.

Desempenho

O tempo de execução usando a máquina original com a melhoria (te_{novo}) será igual ao tempo gasto usando a parte da máquina sem melhoria mais o tempo gasto usando a melhoria:

$$te_{novo} = te_{antigo} \times \left((1 - F_{melhoria}) + \frac{F_{melhoria}}{S_{melhoria}} \right)$$

O *speed up* total (S_{total}) é a razão entre os tempos de execução:

$$S_{total} = \frac{te_{antigo}}{te_{novo}} = \frac{1}{(1 - F_{melhoria}) + \frac{F_{melhoria}}{S_{melhoria}}}$$

Desempenho

Ex: Suponha uma melhoria que execute 10 vezes mais rápido do que o original, mas é utilizada somente 40% do tempo.

$$F_{melhoria} = 0.4$$

$$S_{melhoria} = 10$$

$$\Rightarrow S_{total} = \frac{1}{0.6 + \frac{0.4}{10}} = \frac{1}{0.64} \approx 1.56$$

A lei de Amdahl serve como um indicativo de quanto uma melhoria irá aumentar o desempenho total e como distribuir recursos para melhorar a relação custo/desempenho.

O objetivo é investir recursos proporcionalmente aonde o tempo é gasto.

Desempenho

Ex: Suponha que a operação de raiz quadrada em ponto-flutuante (*rqpf*) é responsável por 20% do tempo de execução. Uma implementação em *hardware* dessa operação irá torná-la 10 vezes mais rápida. Por outro lado, as instruções de ponto-flutuante (*pf*) são responsáveis por 50% do tempo de execução e podem ser melhoradas, a fim de serem executadas 2 vezes mais rápido.

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

$$S_{pf} = \frac{1}{(1 - 0.5) + \frac{0.5}{2.0}} = \frac{1}{0.75} = 1.33$$

Desempenho

O tempo de CPU (tc) para um programa pode ser expresso pelo número de ciclos de *clock* (ncc) da CPU e pelo tempo de ciclo de *clock* (cc):

$$tc = ncc \times cc$$

$$tc = \frac{ncc}{f}$$

Outra métrica é o número médio de ciclos de *clock* por instrução (cci), que depende do número de instruções executadas (ni) e do número de ciclos de *clock* utilizados (ncc):

$$cci = \frac{ncc}{ni}$$

Desempenho

Pode-se obter o tempo de CPU (tc) em função do número de instruções executadas (ni) e do número médio de ciclos de *clock* por instrução (cci):

$$tc = ni \times cci \times cc \quad \Rightarrow \quad tc = \frac{ni \times cci}{f}$$

O desempenho da CPU depende, então, de três fatores: tempo de ciclo de *clock* (cc), ciclos de *clock* por instrução (cci) e número de instruções executadas (ni). Cada um desses fatores está associado a uma tecnologia:

- cc : tecnologia de *hardware* e organização
- cci : organização e conjunto de instruções
- ni : conjunto de instruções e compilador

Desempenho

O número de ciclos de *clock* (ncc) de CPU pode ser expresso pelo número de vezes que a instrução i é executada no programa (ni_i) e pelo número médio de ciclos de *clock* para a instrução i (cci_i):

$$ncc = \sum_{i=1}^n cci_i \times ni_i$$

Logo, o tempo de CPU pode ser expresso como:

$$tc = \left(\sum_{i=1}^n cci_i \times ni_i \right) \times cc$$

Desempenho

Da mesma forma, o número médio de ciclos de *clock* por instrução pode ser expresso como:

$$cci = \frac{\sum_{i=1}^n cci_i \times ni_i}{ni} = \sum_{i=1}^n cci_i \times \left(\frac{ni_i}{ni} \right)$$