

Introdução à Arquitetura e Organização de Computadores

Professor: Diego de Assis Monteiro Fernandes

Aula 3 - Desempenho

1 Introdução

Essa aula tem como objetivo demonstrar os meios para se avaliar o desempenho de um computador. Para isso, descreve os fatores de maior destaque que determinam como mensurá-lo. Apesar do desempenho de um computador ser claramente uma questão importante, a seguir são descritos três motivos que ressaltam ainda mais a necessidade de dimensionar o desempenho de uma máquina.

- Entender porque um trecho de software gasta um determinado tempo para executar.
- Porque um conjunto de instruções pode ser implementado de modo a apresentar um melhor desempenho quando comparado a outro.
- Como algumas características de hardware afetam diretamente o desempenho.

1.1 Tempo de resposta e Throughput

Existem duas métricas principais que podem ser utilizadas para medir o desempenho de uma máquina. A seguir estão descritas duas situações que estão muito próximas da definição das duas métricas.

- Se um programa for executado em duas máquinas diferentes, pode-se afirmar que a máquina mais rápida é a que termina antes a execução do programa?
- Se dois computadores estivessem disponíveis para execução de tarefas designadas por vários usuários de uma empresa. Poderia se dizer que o computador mais rápido é o que completa mais tarefas durante um dia?

Como um usuário, tem-se o interesse em reduzir o **tempo de resposta** - o tempo entre o início e o fim de uma tarefa - também conhecido como **tempo de execução**. A questão levantada anteriormente pelo primeiro item está relacionada ao tempo de execução. Já a segunda questão, que envolve uma empresa, está relacionada ao **throughput**, que pode ser definida como a quantidade total de trabalho em um dado tempo. Em resumo, existem duas formas de se avaliar o desempenho de uma máquina: através do tempo de execução ou do throughput.

O escopo dessa aula restringe-se ao primeiro fator - tempo de resposta. A relação entre o tempo de resposta e o desempenho de uma máquina X pode ser considerada pela equação a seguir:

$$Desempenho_X = \frac{1}{Tempo\ De\ Execucao_X} \quad (1)$$

Isso significa que, para duas máquinas X e Y , se o desempenho de X é melhor do que o de Y então $Tempo\ de\ execucao_Y > Tempo\ de\ execucao_X$.

Daí, pode-se dizer através da equação:

$$n = \frac{Desempenho_X}{Desempenho_Y}, \quad (2)$$

Que X é n vezes mais rápida do que Y (Desempenho relativo).

Exemplo:

Se uma máquina A executa um programa em 10 segundos e uma máquina B executa o mesmo programa em 15 segundos, o quanto A é mais rápido do que B?

Sabemos que A é n vezes mais rápido do que B se:

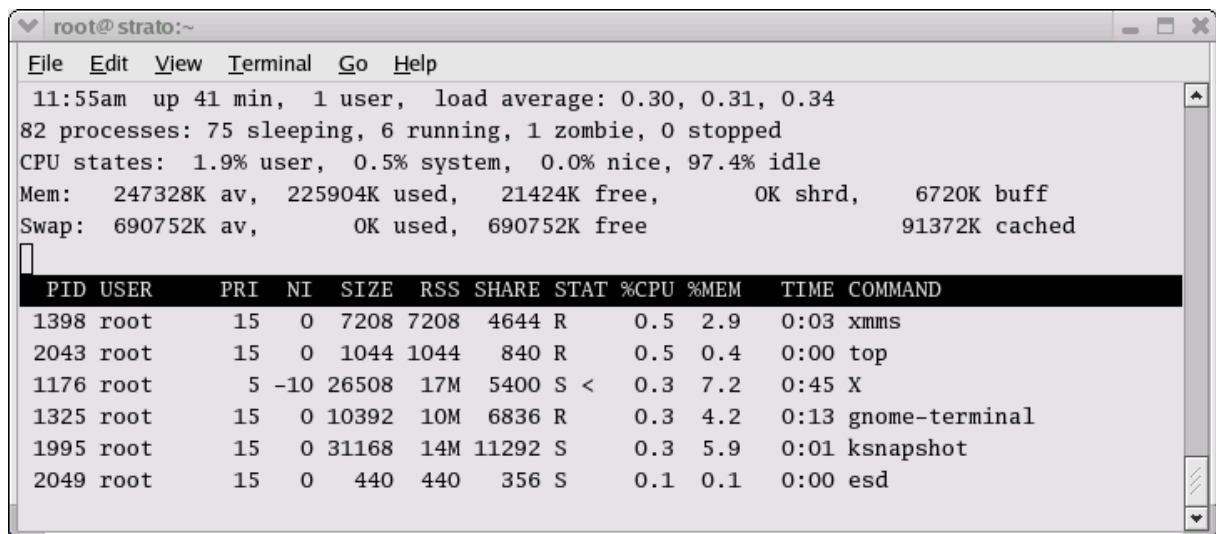
$$n = \frac{\text{Desempenho}_A}{\text{Desempenho}_B} = \frac{\text{Tempo de exec}_B}{\text{Tempo de exec}_A} = \frac{15}{10} = 1,5 \quad (3)$$

Desse modo, A é 1,5 vezes mais rápida do que B.

2 Medindo o Desempenho

Como relatado, o **tempo** é então a medida de desempenho de um computador: o computador que realiza a mesma quantidade de trabalho em menos tempo é o mais rápido. O tempo pode ser definido de diferentes modos, dependendo do que se deseja medir. A definição mais direta é chamada de **tempo de resposta**, que significa o tempo total para realizar uma tarefa, não só a computação das instruções do programa, mas também incluindo acesso a disco, acesso a memória, atividades de E/S, *overhead* do sistema operacional, ...¹

É importante distinguir o tempo transcorrido (tempo de execução) e o tempo em que o processador foi utilizado, ou seja, distinguir entre o tempo do início e fim do programa e a parcela desse tempo em que o processador foi efetivamente utilizado para executar instruções próprias do programa. O **tempo de execução da CPU**, ou simplesmente **tempo de CPU** é o tempo em que o processador gasta computando uma tarefa e não inclui o tempo de espera de uma solicitação a um dispositivo de E/S ou a execução de outras tarefas. O tempo de CPU pode ser ainda dividido entre o tempo em que o processador gasta no programa (chamado **tempo de CPU do usuário**) e o tempo gasto com o sistema operacional em favor do programa (chamado de **tempo de CPU do sistema**).



```
root@strato:~
File Edit View Terminal Go Help
11:55am up 41 min, 1 user, load average: 0.30, 0.31, 0.34
82 processes: 75 sleeping, 6 running, 1 zombie, 0 stopped
CPU states: 1.9% user, 0.5% system, 0.0% nice, 97.4% idle
Mem: 247328K av, 225904K used, 21424K free, 0K shrd, 6720K buff
Swap: 690752K av, 0K used, 690752K free, 91372K cached

  PID USER   PRI NI   SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 1398 root    15  0   7208  7208  4644 R    0.5  2.9   0:03 xmms
 2043 root    15  0   1044  1044   840 R    0.5  0.4   0:00 top
 1176 root     5 -10 26508  17M  5400 S <   0.3  7.2   0:45 X
 1325 root    15  0  10392  10M  6836 R    0.3  4.2   0:13 gnome-terminal
 1995 root    15  0  31168  14M 11292 S    0.3  5.9   0:01 ksnapshot
 2049 root    15  0    440   440   356 S    0.1  0.1   0:00 esd
```

Figura 1: Aplicativo *top* do Linux. Na porção superior da figura pode-se observar os dados referentes ao consumo do processador (cpu states) pelo usuário (user), sistema (system) e o tempo ocioso (idle).

¹Os computadores geralmente podem executar diversos programas simultaneamente. Esse esquema é chamado de multi-tarefa e é provido pelo sistema operacional. Nesses casos, o sistema pode tentar otimizar o *throughput* ao invés de minimizar o tempo de execução.

```

root@strato:~/diego
File Edit View Terminal Go Help
root@strato:~/diego
[root@strato diego]# time ps -ef > tmp
real    0m0.254s
user    0m0.014s
sys     0m0.027s
[root@strato diego]#

```

Figura 2: Exemplo do aplicativo *time* do Linux. O aplicativo, invocado em um shell, mostra o tempo total de execução, o tempo de CPU do usuário e o tempo de CPU do sistema gasto para executar `ps -ef > tmp`. É possível perceber que o tempo total é bem maior do que a soma dos

Considerar somente o tempo de CPU do usuário e ignorar o tempo de CPU do sistema para medir o desempenho pode não ser uma boa idéia. A justificativa para isso é a de que, nos casos onde máquinas executando sistemas operacionais diferentes forem comparadas, o resultado da comparação pode não ser justo quando somente o tempo de CPU do usuário é considerado. A injustiça nesse caso é justificada pelo fato de que uma tarefa em um dos sistemas operacionais pode contribuir para o tempo do usuário enquanto que a mesma tarefa pode contribuir no tempo do sistema no outro sistema operacional. Em resumo, para avaliar o desempenho deve-se considerar somente o tempo de CPU (e não o tempo total de execução), incluindo tanto a porção de usuário e de sistema.

2.1 Clock

Como já foi dito, o tempo de execução é fundamental para um usuário de computador e é, por isso, a métrica fundamental para avaliar o desempenho. Entretanto, quando examinamos os detalhes de uma máquina, é conveniente pensar em desempenho através de outras métricas. Em particular, projetistas de computadores desejam saber o quão rápido um computador realiza operações básicas. Para sincronizar tais operações, quase todos os computadores são construídos com um *clock* que determina quando os eventos acontecem no *hardware*. O *clock* (relógio) envia sinais periódicos para todo o computador. Esses intervalos discretos são chamados de **ciclos de clock** (ou *ticks*). O **período de clock** é o tempo passado em um ciclo (por exemplo, 2 nanosegundos), ou seja, o intervalo entre dois sinais de *clock* e a **taxa de clock** (500 MHz ou mega hertz) é o inverso de um período de clock que indica quantos ciclos acontecem durante um segundo (500MHz indicam 500×10^6 ciclos).

3 Relacionando as Métricas

Usuários e projetistas geralmente examinam o desempenho utilizando métricas diferentes. Se tais métricas forem relacionadas, então pode-se determinar com mais facilidade os efeitos de uma mudança no projeto de um computador na visão do usuário. A seguinte fórmula relaciona as métricas básicas já descritas (ciclo e tempo de ciclo) ao tempo de CPU.

$$\text{Tempo de CPU} = \text{ciclos gastos no programa} \times \text{tempo de ciclo} \quad (4)$$

Alternativamente, considerando que taxa de clock e tempo de ciclo são inversos .

$$\text{Tempo de CPU} = \frac{\text{Ciclos gastos no programa}}{\text{Taxa de clock}} \quad (5)$$

Essa fórmula deixa claro que um projetista de hardware pode melhorar o desempenho reduzindo o tempo de ciclo ou o número de ciclos requerido por um programa.

Exemplo:

Um programa executa em 10secs no computador A , que tem um clock de 400Mhz. Um projetista de hardware deseja construir uma máquina B que executa este programa em 6secs. O projetista determinou que um aumento na taxa de clock é possível, mas este aumento afeta o resto do projeto, fazendo com que a máquina B gaste 1,2 vezes ciclos a mais para este programa. Qual a taxa de clock que o projetista deve buscar?

O tempo de CPU gasto na máquina A :

$$\text{Tempo de CPU} = \frac{\text{Ciclos gastos no programa}}{\text{Taxa de clock}} \Rightarrow 10\text{secs} = \frac{\text{ciclos gastos}_A}{400 \times 10^6} \Rightarrow \quad (6)$$

$$\text{ciclos gastos}_A = 10\text{secs} \times 400 \times 10^6 = 4000 \times 10^6 \text{ciclos} \quad (7)$$

O tempo de CPU gasto na máquina B :

$$\text{Tempo de CPU} = \frac{1,2 \times \text{Ciclos gastos}_A}{\text{Taxa de clock}_B} \Rightarrow 6\text{secs} = \frac{1,2 \times 4000 \times 10^6}{\text{Taxa de clock}_B} \Rightarrow \quad (8)$$

$$\text{Taxa de clock}_B = \frac{1,2 \times 4000 \times 10^6}{6\text{secs}} = 800\text{MHz} \quad (9)$$

A máquina B deve ter o dobro da taxa de clock de A para executar o programa em 6 segundos.

As equações apresentadas não incluem quaisquer referências ao número de instruções necessárias para o programa. Entretanto, uma vez que o compilador gera as instruções para serem executadas e a máquina necessita executá-las para executar o programa, é fundamental que o tempo de execução também dependa diretamente do número de instruções do programa. A equação a seguir indica onde a quantidade de instruções de um programa pode ser incluída para contribuir no cálculo do tempo de execução.

$$\text{Ciclos de clock da CPU} = \text{Instrucoes do programa} \times \text{quantidade media de ciclos por instrucao} \quad (10)$$

Onde, o termo ciclo por instrução, que é o número médio de ciclos em que cada instrução leva para executar, é geralmente abreviada por **CPI**. Uma vez que cada instrução pode gastar uma quantidade diferente de ciclos para executar, pois cada instrução executa uma tarefa específica, CPI é a média de todas as instruções executadas no programa. CPI provê um meio de comparar duas implementações diferentes de um mesmo conjunto de instruções, uma vez que as instruções de um programa serão as mesmas.

Exemplo: Suponha que existam duas implementações de um mesmo conjunto de instruções: as máquinas A e B . A máquina A tem o tempo de um ciclo de 1ns e CPI de 2,0 para um determinado programa. A máquina B tem o tempo de ciclo de 2ns e CPI de 1,2 para o mesmo programa. Determine que máquina é mais rápida para esse programa e o quanto mais rápida é?

As duas máquinas executam o mesmo número de instruções. Vamos chamar esse número de I . A quantidade de ciclos gastos para as duas máquinas é:

$$\text{Ciclos}_A = I \times 2,0$$

$$\text{Ciclos}_B = I \times 1,2$$

Agora podemos computar o tempo de CPU para cada máquina:

$$\text{Tempo}_A = \text{Ciclos}_A \times \text{Tempo de Ciclo}_A = I \times 2,0 \times 1\text{ns} = 2 \times \text{Ins}$$

$$\text{Tempo}_B = \text{Ciclos}_B \times \text{Tempo de Ciclo}_B = I \times 1,2 \times 2\text{ns} = 2,4 \times \text{Ins}$$

Claramente a máquina A é mais rápida. A proporção é a seguinte:

$$\frac{\text{Desempenho de CPU}_A}{\text{Desempenho de CPU}_B} = \frac{\text{Tempo de B}}{\text{Tempo de A}} = \frac{2,4 \times \text{Ins}}{2 \times \text{Ins}} = 1,2 \quad (11)$$

Baseado nas métricas apresentadas, pode-se então escrever a fórmula do tempo de outro modo:

$$\text{Tempo de CPU} = \text{Qtde de Instrucoes} \times \text{CPI} \times \text{tempo de ciclo} \quad (12)$$

$$\text{TempodeCPU} = \frac{\text{Qtde de Instrucoes} \times \text{CPI}}{\text{TaxadeClock}} \quad (13)$$

Componentes de Desempenho	Unidade de medida
Tempo de CPU de um programa	Segundos por programa
Quantidade de instruções	Número de instr. executadas pelo programa
CPI	Número médio de ciclos por instrução
Tempo de ciclo	Segundos por ciclo

Tabela 1: Métricas de desempenho e suas respectivas unidades de medida.

Exemplo:

Um projetista de compilador está tentando decidir entre duas sequências de código para uma máquina em particular. O projetista de hardware forneceu os seguintes dados:

Classe de instrução	CPI da classe
A	1
B	2
C	3

Para uma linguagem de alto nível, o projetista de compilador está considerando duas sequências de código que requerem a seguinte quantidade de instruções:

Sequência de Código	Quantidade de Instruções		
	A	B	C
1	2	1	2
2	4	1	1

Que sequência de código executa mais instruções? Qual será mais rápido? Qual é a CPI para cada sequência?

A sequência 1 executa $2 + 1 + 2 = 5$ instruções. A sequência 2 executa $4 + 1 + 1 = 6$. Logo a sequência 1 executa menos instruções.

Para calcular o número de ciclos realiza-se o seguinte:

$$\text{CiclosdeCPU}_1 = (2 \times 1) + (1 \times 2) + (2 \times 3) = 2 + 2 + 6 = 10 \text{ ciclos}$$

$$\text{CiclosdeCPU}_2 = (4 \times 1) + (1 \times 2) + (1 \times 3) = 4 + 2 + 3 = 9 \text{ ciclos}$$

Portanto a sequência 2 é mais rápida, embora execute um número maior de instruções. O valor de CPI para as sequências será:

$$\text{CPI}_1 = \frac{\text{Ciclos}}{\text{QtdeInst}} = \frac{10}{5} = 2$$

$$\text{CPI}_2 = \frac{\text{Ciclos}}{\text{QtdeInst}} = \frac{9}{6} = 1,5$$

4 MIPS (Million Instructions per second)

MIPS foi uma métrica utilizada para medir o desempenho de computadores. Essa métrica determina simplesmente uma taxa de execução de instruções durante um segundo. Para um determinado programa, MIPS pode ser calculado pela equação mostrada a seguir:

$$\text{MIPS} = \frac{\text{quantidade de instrucoes}}{\text{Tempo de Exec} \times 10^6} \quad (14)$$

Apesar dessa métrica ter sido utilizada para avaliar o desempenho, será que realmente ela deve ser utilizada com tal intuito? Em outras palavras, MIPS é uma medida de desempenho justa? A resposta é não, que pode ser justificada por três pontos fundamentais.

- MIPS especifica a taxa de execução de instruções, mas não leva em conta as capacidades das instruções. Como comparar arquiteturas diferentes através do MIPS, ou seja, como comparar conjunto de instruções diferentes com MIPS? Uma vez em que as arquiteturas são distintas, as instruções realizar operações diferentes e esse tipo de comparação não é interessante.
- MIPS pode variar entre programas no mesmo computador, uma vez que cada programa executa instruções diferentes e quantidades diferentes de instruções. Isso permite que diferentes taxas possam ser calculadas para uma mesma arquitetura.
- MIPS pode variar inversamente com o desempenho, ou seja, a comparação entre taxas de MIPS e o desempenho avaliado de acordo com as equações descritas durante a aula podem indicar resultados contraditórios.

O exemplo a seguir demonstra os tópicos descritos, reforçando a idéia de que não é interessante utilizar MIPS para avaliar o desempenho.

Exemplo:

Considere uma máquina com três classes de instruções com as seguintes CPI.

Classe de instrução	CPI da classe
A	1
B	2
C	3

Agora supondo que deseja-se medir o código para o mesmo programa a partir de dois compiladores diferentes. Os seguintes resultados foram obtidos:

Código	Quantidade de Instruções(em bilhões)		
	A	B	C
Compilador 1	5	1	1
Compilador 2	10	1	1

Assumindo que o clock da máquina é de $500MHz$. Qual sequência executará mais rápido de acordo com MIPS? E de acordo com o tempo de execução?

Calculando o número de ciclos:

$$Ciclos_1 = (5 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 10 \times 10^9 \text{ ciclos}$$

$$Ciclos_2 = (10 \times 1 + 1 \times 2 + 1 \times 3) \times 10^9 = 15 \times 10^9 \text{ ciclos}$$

Encontrando o tempo de execução:

$$Tempo_1 = \frac{10 \times 10^9}{500 \times 10^6} = 20 \text{ segundos}$$

$$Tempo_2 = \frac{15 \times 10^9}{500 \times 10^6} = 30 \text{ segundos}$$

De acordo com o tempo de execução, o compilador 1 gerou o programa mais rápido. Calculando agora a taxa de MIPS para cada versão do programa:

$$MIPS_1 = \frac{\text{numero de instrucoes}}{\text{Tempo de Exec} \times 10^6} = \frac{(5+1+1) \times 10^9}{20 \times 10^6} = 350$$

$$MIPS_2 = \frac{\text{numero de instrucoes}}{\text{Tempo de Exec} \times 10^6} = \frac{(10+1+1) \times 10^9}{30 \times 10^6} = 400$$

O código gerado pelo compilador 2 tem uma taxa de MIPS mais alta, entretanto o código do compilador 1 executa mais rápido!

5 Exercícios

1 - Desejamos comparar o desempenho de duas máquinas diferentes: M1 e M2. As seguintes medidas foram feitas nessas máquinas:

Programa	Tempo em M1	Tempo em M2
1	10 segundos	5 segundos
2	3 segundos	4 segundos

Qual máquina é mais rápida em cada programa e por quanto?

2- Considere as duas máquinas e programas do exercício anterior. As seguintes medidas adicionais foram feitas:

Programa	Instruções exec. em M1	Instruções exec. em M2
1	200×10^6	160×10^6

Encontre a taxa de execução de instruções (instruções por segundo) para cada máquina executando o programa 1.

3 - Se a taxa de clock das máquinas M1 e M2 do exercício 1 é de 200MHz e 300MHz, respectivamente, encontre o CPI para o programa 1 em ambas as máquinas.

4 - Assumindo que o CPI do programa 2 em cada máquina do exercício 1 é o mesmo para o programa 1 encontrado no exercício 3, encontre a quantidade de instruções para o programa 2 executando em cada máquina usando os tempos do exercício 1.