

02 - Avaliação do Desempenho

Luís Paulo Santos
Arquitectura de Computadores 2014/15
Universidade do Minho

Desempenho do CPU

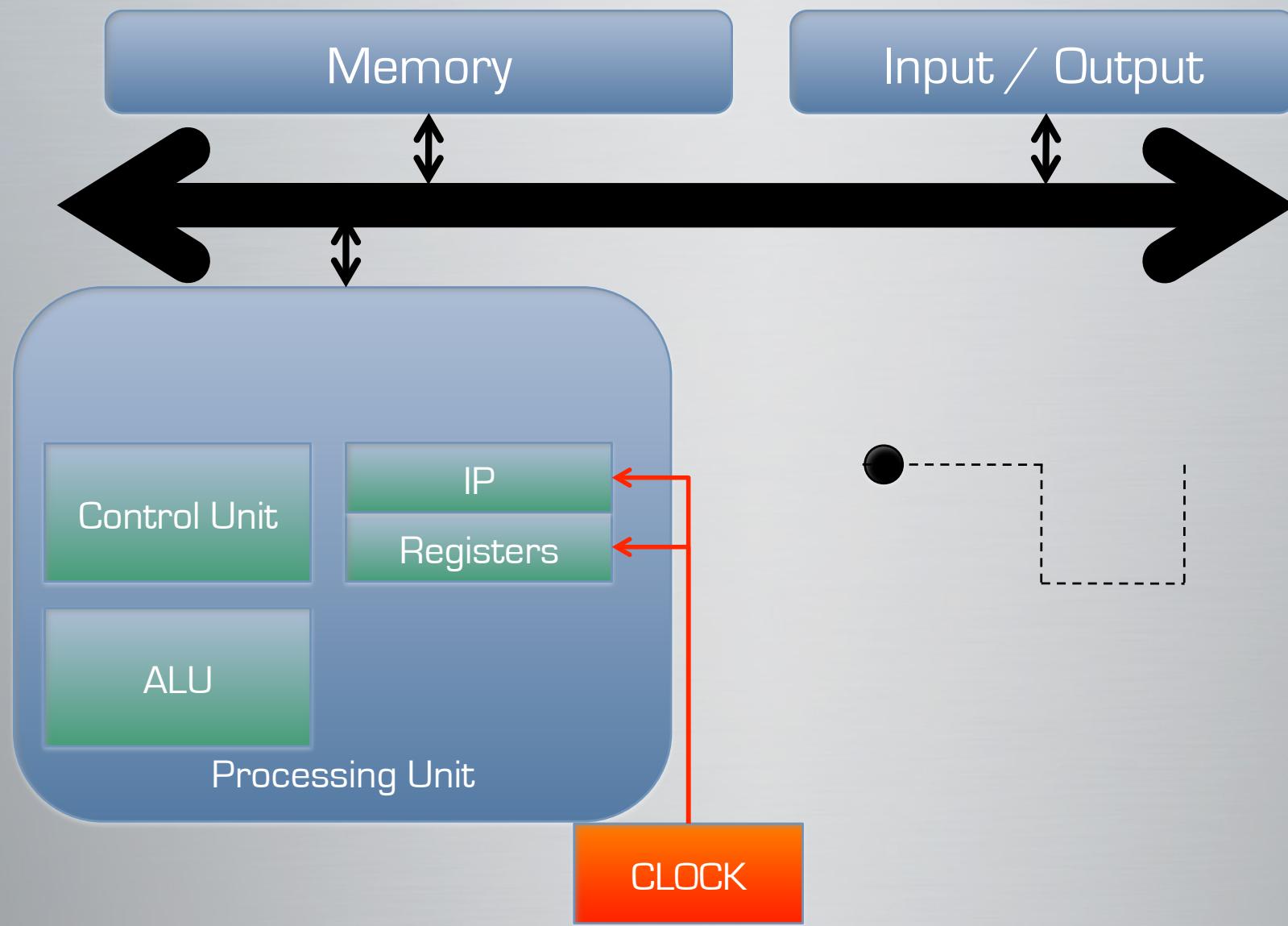
Previsão do tempo de execução (T_{EXEC}) de um programa numa máquina - requer um **modelo** que relacione o desempenho com as características do sistema de computação ($hw + sw$)

Um programa numa máquina executa num determinado número médio de ciclos de relógio:
clock cycles

O período do relógio do CPU é constante:
Tcc

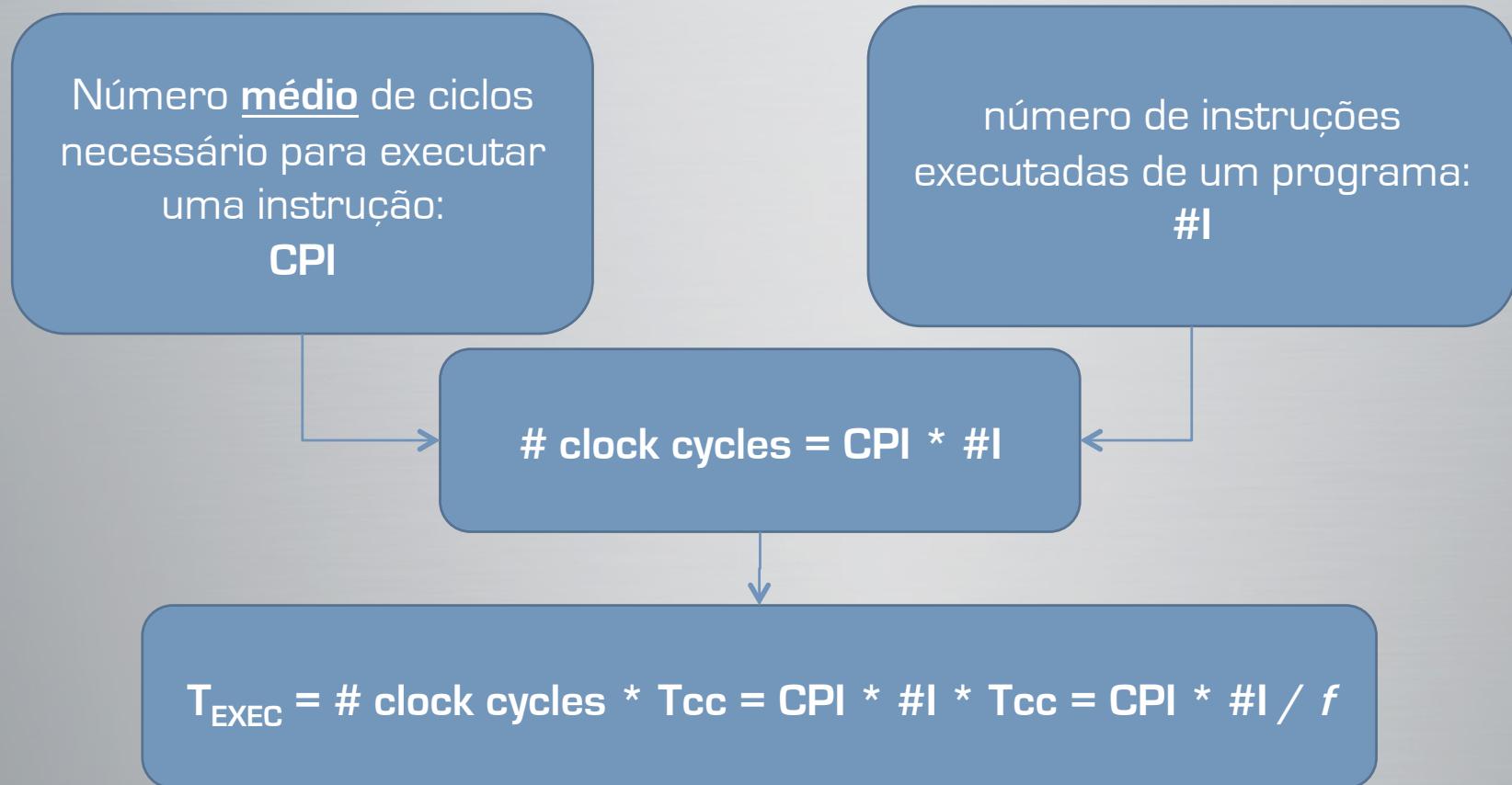
$$T_{EXEC} = \# \text{ clock cycles} * Tcc$$

Avaliação Desempenho



Desempenho do CPU

- De que depende o número médio de ciclos necessários para executar um programa?



Desempenho do CPU

$$T_{EXEC} = CPI * \#I / f$$

- CPI é um valor médio, logo pode ser medido com diferentes precisões.
- A aproximação mais grosseira será medir o CPI, independentemente do tipo de instruções.
- Diferentes tipos de instruções exibem diferentes CPI:
 - CPI Divisões > CPI multiplicações
 - Acessos à memória exigem mais ciclos do que acessos a registos
 - Operações em vírgula flutuante podem exigir mais ciclos do que operações com inteiros

Desempenho do CPU

- Um programador quer escolher entre dois segmentos de código diferentes para um mesmo algoritmo. Qual o mais rápido?

Tipo de Instrução	CPI
A	1
B	2
C	3

Código	Número de Instruções		
	A	B	C
1	2000	1000	100
2	100	1000	1000

$$T_{EXEC1} = (1*2000 + 2*1000 + 3*100) / f = 4300 / f$$

$$T_{EXEC2} = (1*100 + 2*1000 + 3*1000) / f = 5100 / f$$

$$Ganho = \frac{T_{EXEC2}}{T_{EXEC1}} = \frac{5100}{4300} = 1,186$$

Desempenho do CPU

- Calcule o tempo de execução do programa abaixo numa máquina com um relógio de 2 GHz e CPI=1.5

```
    movl 10, %eax
    movl 0, %ecx
ciclo:
    addl %eax, %ecx
    decl %eax
    jnz ciclo
```

#I = 32

NOTA: número de instruções **executadas**.

$$T_{exec} = 32 * 1.5 / 2E9 = 24E-9 \text{ s} = 24 \text{ ns}$$

Relação entre as métricas

$$T_{EXEC} = CPI * \#I / f$$

- $\#I$ – depende do algoritmo, do compilador e da arquitectura (ISA)
- CPI – depende da arquitectura (ISA), da mistura de instruções efectivamente utilizadas, da organização do processador e da organização dos restantes componentes do sistema (ex., memória)
- f – depende da organização do processador e da tecnologia utilizada

“A única métrica completa e fiável para avaliar o desempenho de um computador é o tempo de execução”

As métricas CPI, f e $\#I$ não podem ser avaliadas isoladamente, devendo ser sempre consideradas em conjunto, pois dependem umas das outras.

Relação entre as métricas

Exemplo 1 : Aumentar f (diminuir Tcc) implica frequentemente um aumento do CPI!

Explicação: Se Tcc diminui, mas o tempo de acesso à memória (Tmem) se mantém, são necessários mais ciclos para aceder à memória.

$$f_1 = 1GHz$$

$$T_{cc1} = 1ns$$

$$T_{mem} = 40ns$$

$$Ciclos_{mem1} = 40$$

$$f_2 = 2GHz$$

$$T_{cc2} = 0.5ns$$

$$T_{mem} = 40ns$$

$$Ciclos_{mem2} = 80$$

Conclusão: Apesar de Tcc diminuir para metade, Texec não diminui para metade, pois o número de ciclos de acesso à memória aumenta.

Relação entre as métricas

Exemplo 2 : Diminuir o número de instruções (#I) recorrendo a instruções mais complexas resulta num aumento do CPI!

Explicação: As instruções mais complexas realizam o trabalho de várias instruções simples, mas podem necessitar de mais ciclos para o completar, resultando num aumento do CPI.

Este é um dos argumentos dos defensores de arquitecturas RISC.

Conclusão: O número de instruções diminui, mas o ganho em tempo de execução não diminui na mesma proporção, devido ao aumento do CPI.

Desempenho do CPU - MIPS

MIPS (milhões de instruções por segundo) – uma métrica enganadora

$$\text{MIPS nativo} = \frac{\# I}{T_{exec} * 10^6}$$

1. MIPS especifica a taxa de execução das instruções, mas não considera o trabalho feito por cada instrução. CPUs com diferentes *instruction sets* não podem ser comparados.
2. MIPS varia entre diferentes programas no mesmo CPU
3. MIPS pode variar inversamente com o desempenho

Esta métrica pode ser usada para comparar o desempenho do mesmo programa em CPUs com o mesmo conjunto de instruções, mas micro-arquitecturas e/ou frequências do relógio diferentes.

Desempenho do CPU - MIPS

- Considere os seguintes segmentos de código executados numa máquina com $f = 1$ GHz. Qual o que exibe melhor desempenho de acordo com as métricas Texec e MIPS?

Código	Número de Instruções		
	A (CPI=1)	B (CPI=2)	C (CPI=3)
1	5	1	1
2	10	1	1

$$T_{exec1} = \frac{5 + 2 + 3}{10^9} = 10ns$$

$$T_{exec2} = \frac{10 + 2 + 3}{10^9} = 15ns$$

$$MIPS_1 = \frac{7}{10 * 10^{-9} * 10^6} = 700$$

$$MIPS_2 = \frac{12}{15 * 10^{-9} * 10^6} = 800$$

Esta métrica favorece programas com muitas instruções simples e rápidas, pois não tem em consideração a quantidade de trabalho feita por cada uma.

Desempenho do CPU - MIPS

MIPS de pico- máxima taxa **possível** de execução de instruções

É a métrica mais enganadora, pois corresponde a sequências de código que apenas tenham instruções com o CPI mais baixo possível. Este tipo de sequências de instruções não realizam, regra geral, trabalho útil; consistem apenas em operações elementares com operandos em registos.

Pode ser visto como “a velocidade da luz” do CPU, e portanto, inatingível.

O principal problema é que é muitas vezes publicitada pelos fabricantes/vendedores como uma medida de desempenho das suas máquinas!

Desempenho - CPE

- As métricas CPI e MIPS dependem do número de instruções máquina efectivamente executadas
- Um programador de uma linguagem de alto nível não necessita de métricas mais próximas do problema que se pretende resolver
- **CPE – Ciclos Por Elemento**

“número médio de ciclos necessários para processar um elemento de dados”

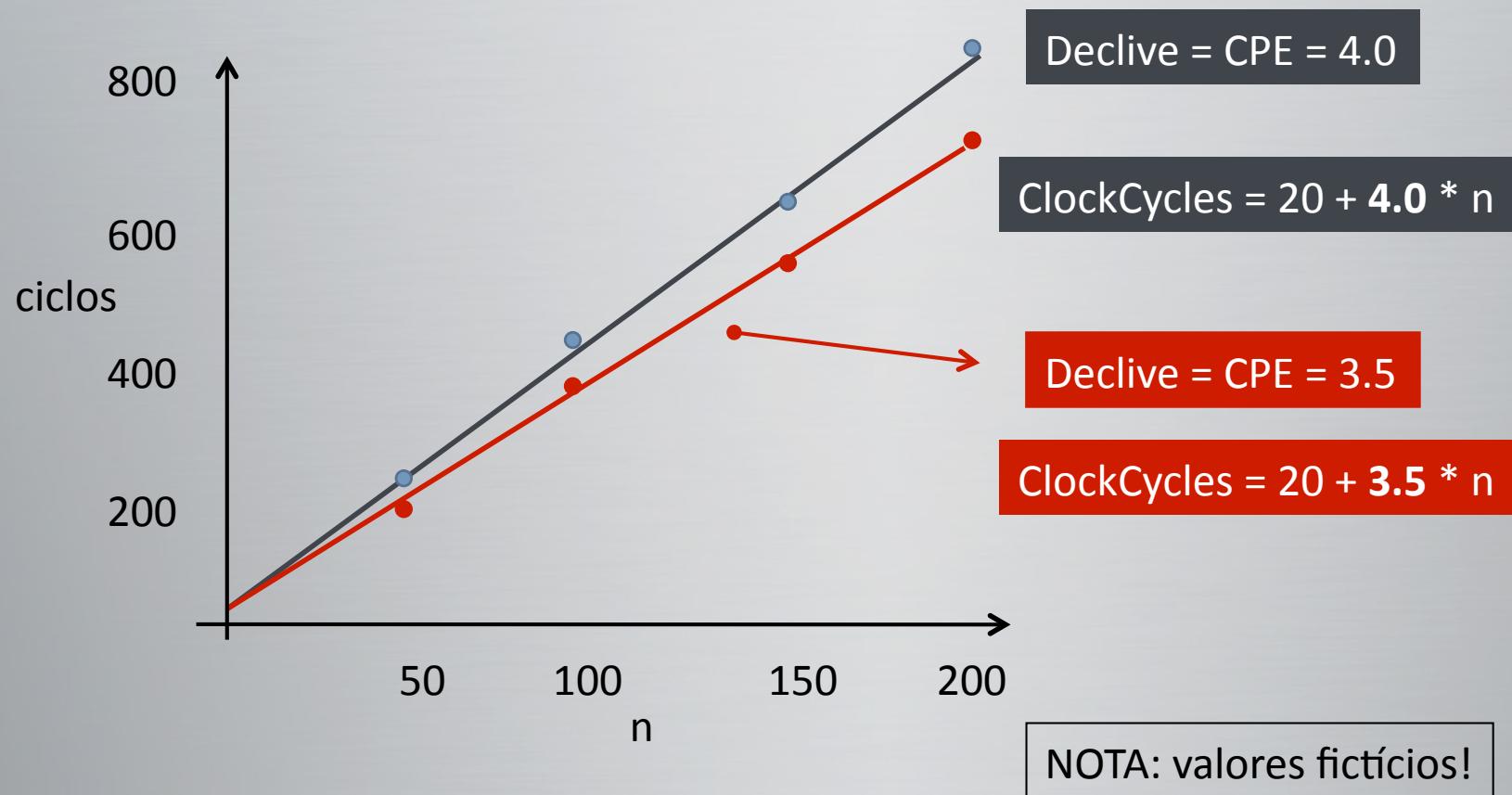
Ajuda a perceber o desempenho do ciclo de um programa iterativo.

Apropriada para expressar o desempenho de um programa que realiza uma operação repetitiva sobre diferentes elementos de dados:

Desempenho - CPE

```
void metade1 (int *a, int n) {  
    for (int i=0 ; i<n ; i++)  
        a[i] = a[i] /2;  
}
```

```
void metade2 (int *a, int n) {  
    for (int i=0 ; i<n ; i++)  
        a[i] >>= 1;  
}
```



Desempenho - CPE

```
void metade1 (int *a, int n) {  
    for (int i=0 ; i<n ; i++)  
        a[i] = a[i] /2;  
}
```

Para $n = 1000 \rightarrow$ ciclos = 4020

Qual o CPE?

Quantos ciclos por iteração?

```
void metade3 (int *a, int n) {  
    for (int i=0 ; i<n ; i+=2) {  
        a[i] = a[i] /2;  
        a[i+1] = a[i+1] /2;  
    }  
}
```

Para $n = 1000 \rightarrow$ ciclos = 3820

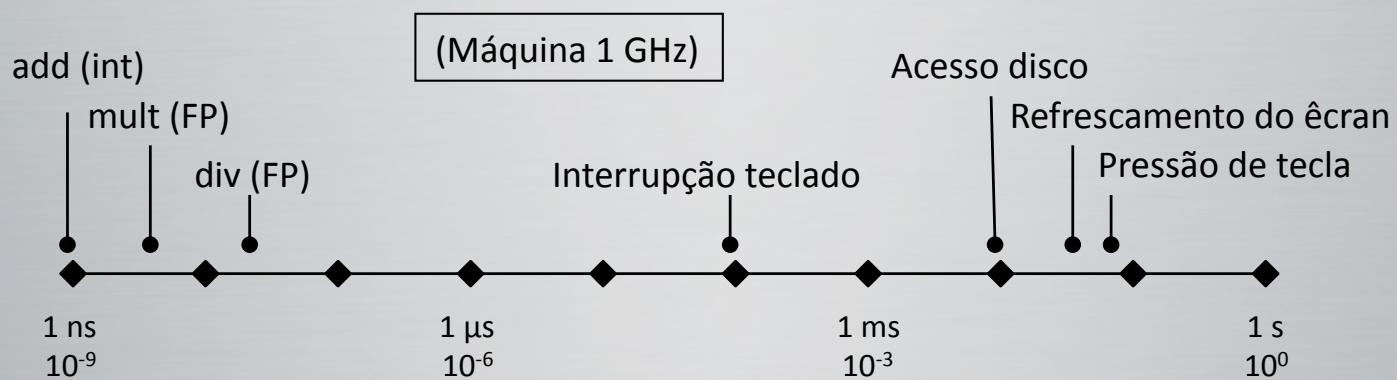
Qual o CPE?

Quantos ciclos por iteração?

A utilização de **ciclos por elemento** e não **ciclos por iteração** dá uma indicação do tempo necessário para processar um vector de tamanho n independentemente da implementação.

Metodologia: Medição de Desempenho

```
before = ReadTimer();  
<<< Code Segment>>>  
after = ReadTimer();  
ElapsedTime = after - before;
```



- **Resolução** do relógio: unidade de tempo entre 2 incrementos do contador
não é possível medir eventos com duração inferior à resolução
- **Precisão** do relógio: diferença entre o valor medido e o tempo efectivamente decorrido

Metodologia: Medição de Desempenho

- Qual o tempo a medir?
 - *Wall Time*
 - Tempo decorrido desde o início até ao fim do programa
 - Depende da carga do sistema (E/S, outros processos,...)
 - Tempo de CPU
 - Tempo efectivamente dedicado a este processo
 - Menos sensível à carga do sistema

Metodologia: Medição de Desempenho

Combinar o resultado de várias medições:

- Média das várias medições
 - Valores muito alto/baixos influenciam a média
 - Analisar também o desvio padrão (e.g., variações entre medições)
- Melhor medição
 - Valor obtido nas condições ideais
- Média das K-melhores medições
 - Média das k melhores execuções vezes
- Mediana
 - Mais robusto a variações nas medições

Metodologia: Medição de Desempenho

Opções para medição do tempo

- `gettimeofday()`
 - Retorna o número de segundos desde 1-Jan-1970
 - Usa “Timer” ou o contador dos ciclos (depende da plataforma)
 - Resolução no melhor caso de 1us
- Contador de ciclos
 - Usa o *time stamp counter* do próprio processador, contando ciclos do relógio
 - Mede o *wall time*
 - Precisão muito elevada
 - Utilizar para medições <<1s
- Time (linha de comando)
 - Apenas usável para medições >>1seg

Metodologia: Medição de Desempenho

Contadores de desempenho

- Lógica incluída nos processadores (modernos) para contagem de eventos específicos
- Atualizados a cada ciclo de relógio
- Vantagens:
 - Não intrusivos / baixa sobrecarga (disponibilizados pelo hardware)
 - Elevada resolução (relógio do processador)
- Pontos fracos:
 - Específicos de cada processador / não existe um “standard”
 - Não são apropriados para serem usados por utilizador “comum”
 - Contabilizados por processador:
 - Sensíveis à carga no sistema / migração de processos entre “cores”

Metodologia: Medição de Desempenho

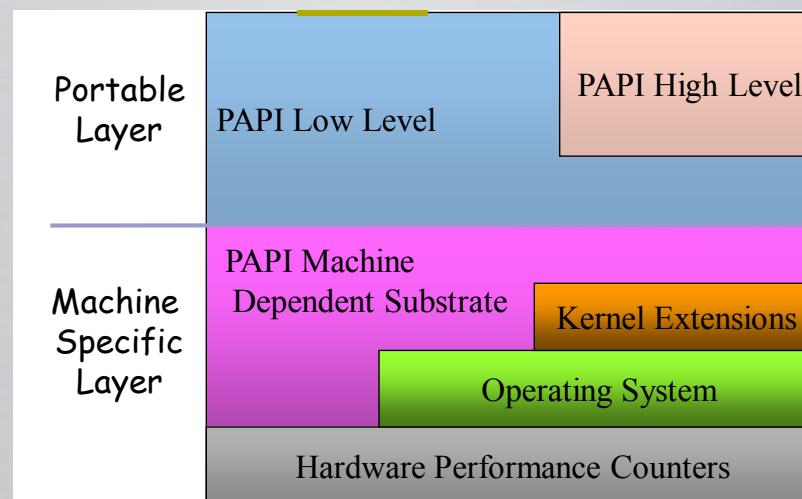
Contadores de desempenho

- Eventos típicos
 - Ciclos de relógio / instruções
 - Instruções de vírgula flutuante
 - Instruções sobre valores inteiros (add, sub, etc)
 - *Load/stores*
 - *Cache misses* (L1, L2, etc)
- Métricas derivadas ajudam a correlacionar o comportamento da aplicação com um ou vários componentes do *hardware*
 - Exemplos: CPI; instruções por *load/store* ; *cache hit rate*
- Valores abaixo de um patamar aceitável numa dessas métricas indicam pontos de potencial optimização

Metodologia: Medição de Desempenho

Performance Application Programming Interface

- Interface para acesso aos contadores de desempenho
- Suporta vários sistemas operativos:
 - Usa o suporte em Linux para acesso aos contadores (kernel posterior a 2.6.31)
- Inclui rotinas para contagem de tempo e para obter informação sobre o sistema
- <http://icl.cs.utk.edu/papi/>
- Arquitetura:



Metodologia: Medição de Desempenho

Performance Application Programming Interface

- Limitações
 - Disponível apenas para alguns SO (Linux com kernel posterior a 2.6.31)
 - É suportado em hardware virtualizado (e.g. VMware) mas as medições são de pior qualidade:
 - A camada de virtualização introduz sobrecarga
 - Não é fácil correlacionar as métricas de baixo nível com características das aplicações

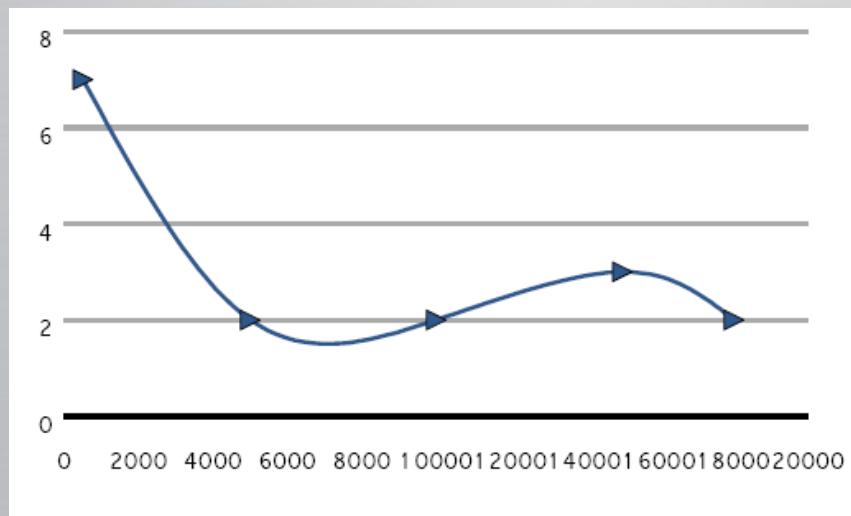
Metodologia: Medição de Desempenho

Apresentação dos resultados

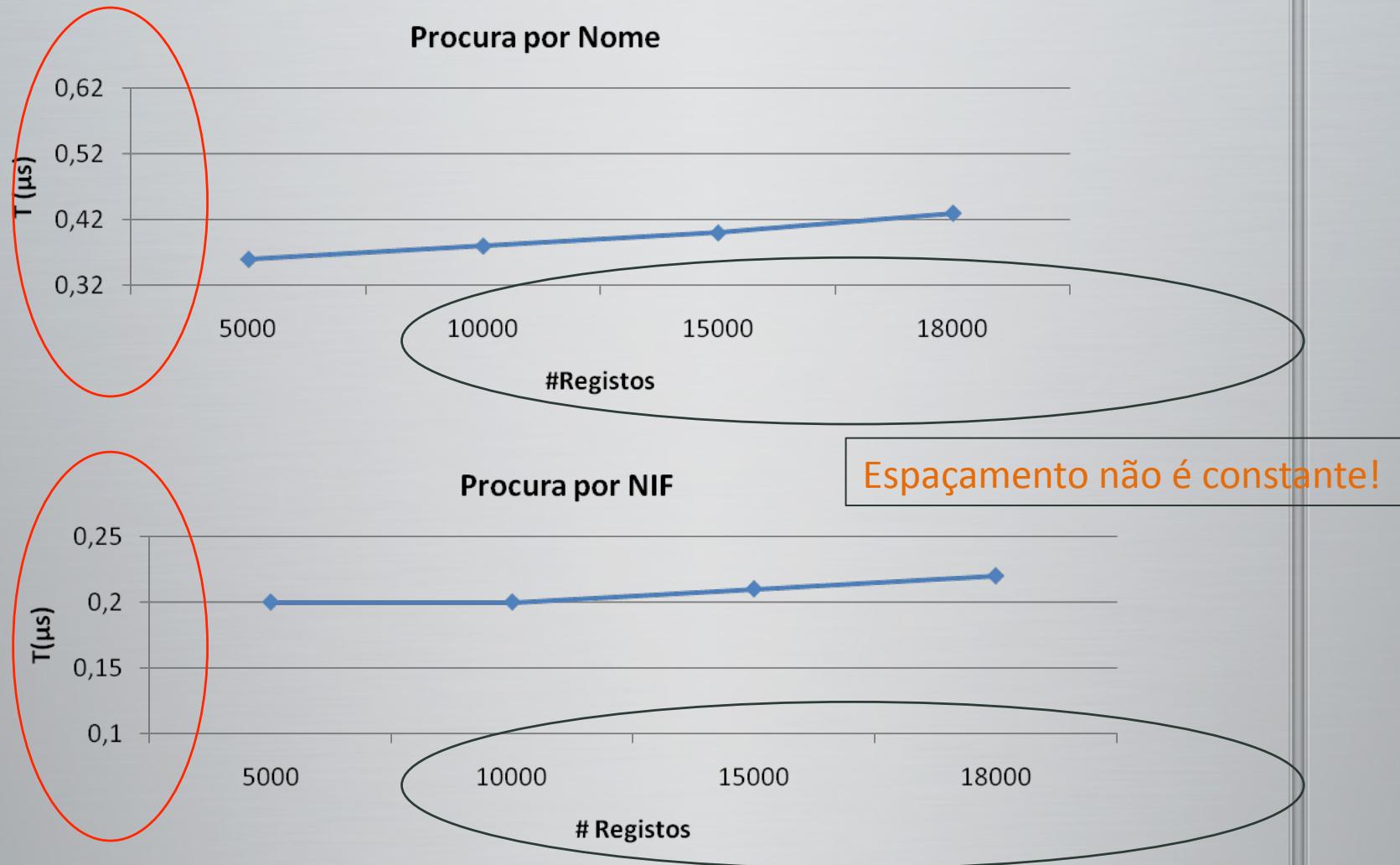
- Apresentar os resultados de forma compacta

Operações	Tempos de Execução			
	5000	10000	15000	18000
Carregar Dados	10.019 ms	20.881 ms	32.027 ms	40.992 ms
Inserir Cliente	7.100 μ s	7.400 μ s	8.800 μ s	9.500 μ s
Procura por Nome	0.360 μ s	0.380 μ s	0.400 μ s	0.430 μ s
Procura por Nif	0.020 μ s	0.020 μ s	0.020 μ s	0.020 μ s
Percorrer Estrutura	0.092 ms	0.232 ms	0.470 ms	0.673 ms

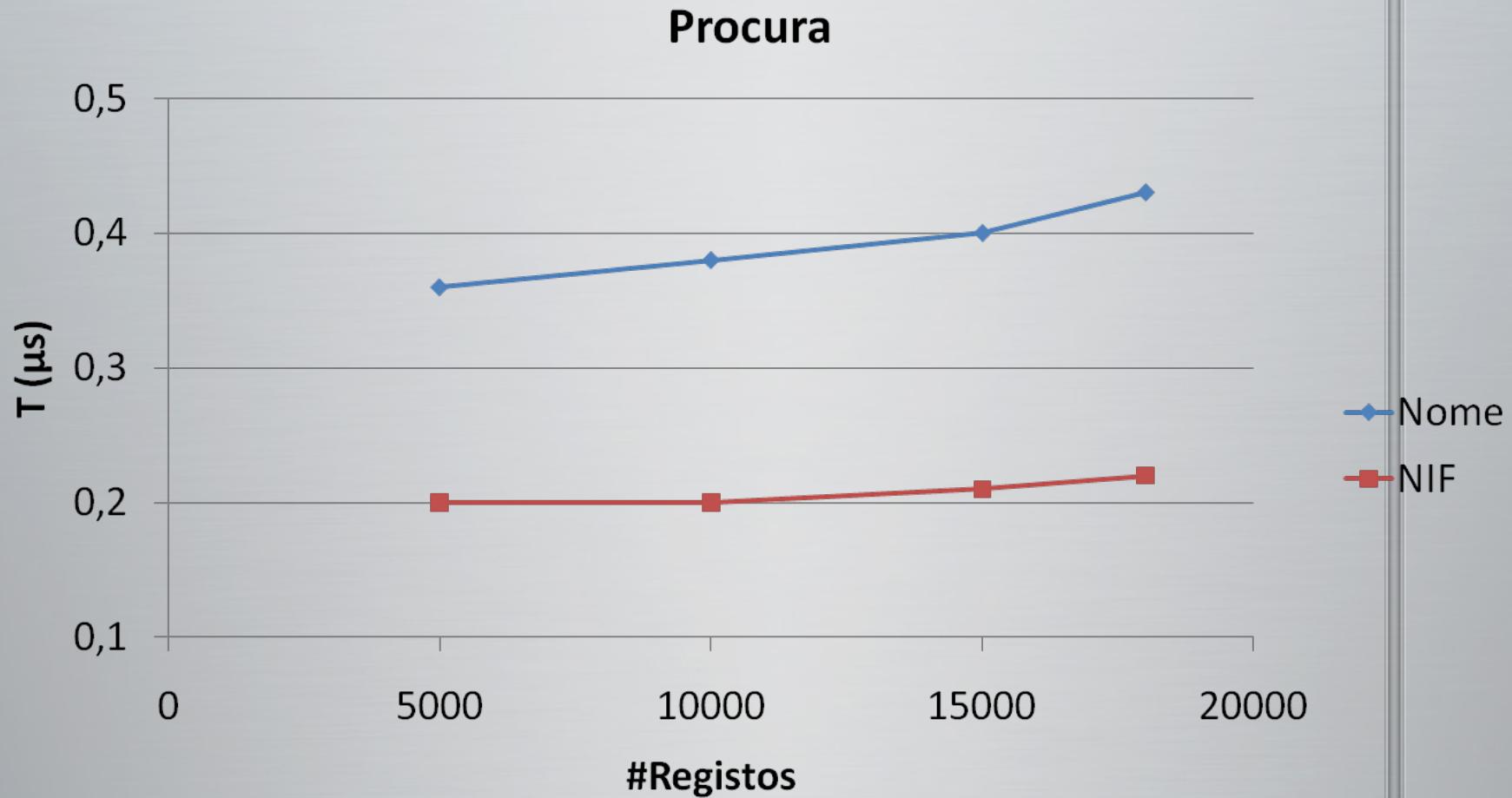
- Colocar legendas nas tabelas e gráficos



Metodologia: Medição de Desempenho

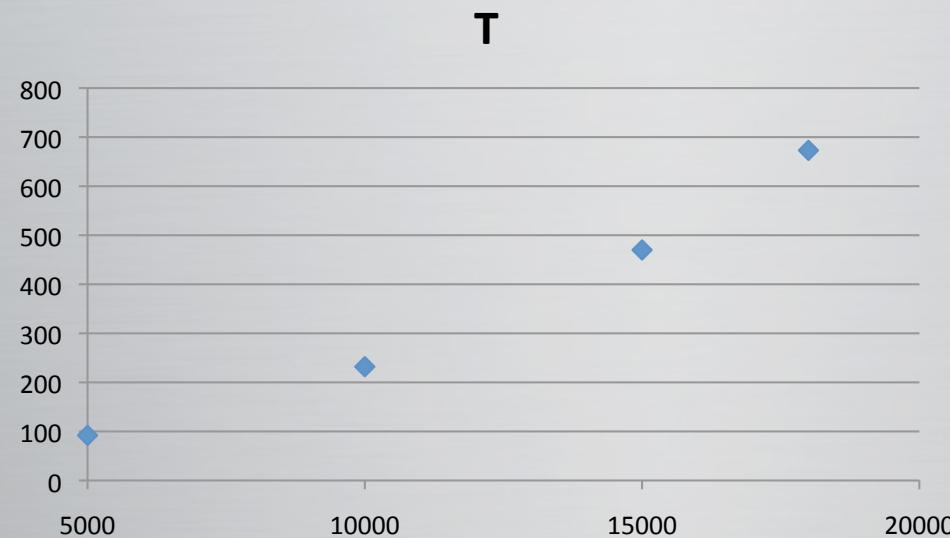


Metodologia: Medição de Desempenho



Complexidade - curve fitting

Tempo de Execução (μs)				
	Nº de registos			
Operação	5000	10000	15000	18000
Percorrer Estrutura	92.00	232.00	470.00	673.00



O processo de curve fitting permite determinar a equação da curva que melhor descreve os dados.

Complexidade – curve fitting

A opção “Trendline” do MS EXCEL determina a equação da curva dado um modelo: linear, polinomial, logarítmico, exponencial, etc.

O parâmetro R2 descreve a qualidade do fitting.

Quanto mais perto de 1 melhor.

