CAP-387(2016) – Tópicos Especiais em Computação Aplicada: Construção de Aplicações Massivamente Paralelas

Aula 26: False-Sharing e Thread-Safety

Celso L. Mendes, Stephan Stephany

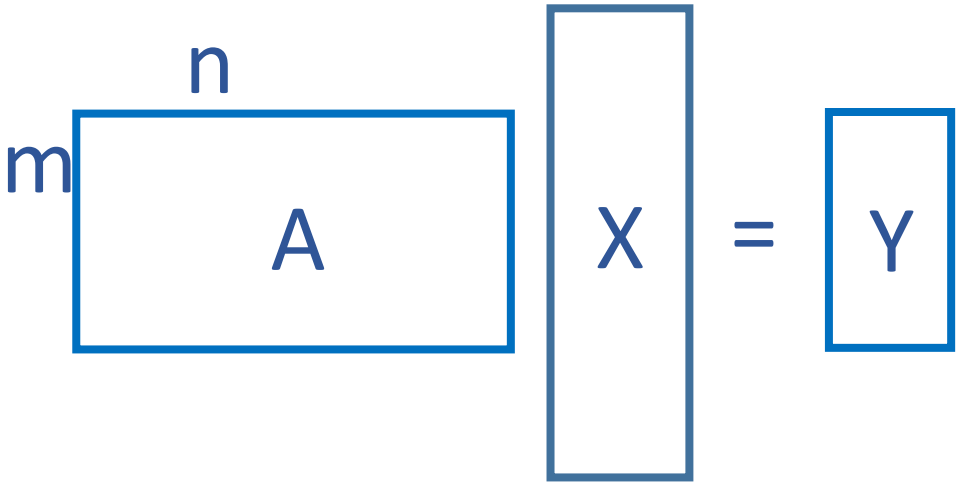
LAC / INPE

Emails: celso.mendes@inpe.br, stephan.stephany@inpe.br

Mais Efeitos: Cache

- Exemplo-2: Multiplicação Matriz × Vetor
  - A(m,n) × X(n) = Y(m)esultados gerados são corretos
  - Supondo valores em ponto-flutuane, precisão dupla
- Código Serial:

```
for (i=0; i<m; i++) {
        y[i]=0.0;
        for (j=0; j<n; j++)
                y[i] += A[i][j] * x[j] ;
}
```



Paralelização com OpenMP

- Paralelização trivial:
  - Não há dependências entre iterações do loop-*i*

```
#pragma omp parallel for private(i,j) shared(A,x,y,m,n)
for (i=0; i<m; i++) {
        y[i]=0.0;
        for (j=0; j<n; j++)
                y[i] += A[i][j] * x[j] ;
}
```

Desempenho Paralelo

- Tempos (em segundos) em algum sistema:

Matriz A → 8.000.000×8 8.000×8.000 8×8.000.000

Threads	Tempo	*Efic.*	Tempo	*Efic.*	Tempo	*Efic.*
1	0,322	*1*	0,264	*1*	0,333	*1*
2	0,219	*0,735*	0,189	*0,698*	0,300	*0,555*
4	0,141	*0,571*	0,119	*0,555*	0,303	***0,275***

Desempenho – Comentários

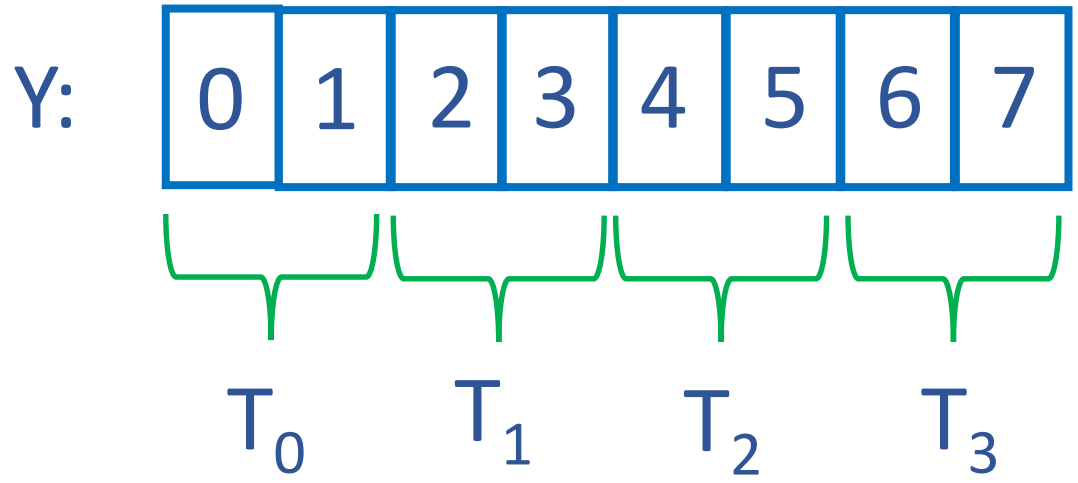
- Número de operações (+ e ×):
  - 64.000.000 em todos os casos
- Execuções com 1 thread
  - Efeitos de cache provavelmente predominam
  - Melhor caso: 8.000×8.000 → ajuste à cache ?
- Execuções *paralelas*:
  - Melhor caso: 8.000×8.000
  - Pior caso: 8×8.000.000

Desempenho – Análise

- Execuções *paralelas*:
  - Pior caso: 8×8.000.000 (m=8, n=8.000.000)
  - Array Y tem apenas 8 elementos!
  - Todos os elementos de Y estão no mesmo bloco de cache! (blocos de 64 bytes)
  - Em toda iteração, threads *escrevem* em algum element de Y

Desempenho – Análise (cont.)

- Execuções *paralelas*:
  - Pior caso: 8×8.000.000 (m=8, n=8.000.000)
  - Array Y tem apenas 8 elementos!
  - Caso de 4 threads:



Desempenho – Análise (cont.)

- Caso: 8×8.000.000 , 4 threads:
  - Cada thread acessa (escreve) exclusivamente dois elementos de Y
  - Porém, todos os elementos estão no mesmo bloco de cache!
    - Fenômeno conhecido como *false-sharing*
  - Cada acesso marca todo o bloco como modificado
  - Bloco precisa ser invalidado na cache!

Thread-Safety

- Definição:
  - Um código é dito *thread-safe* se ele pode ser executado por dois threads simultâneos, e produz resultados corretos
  - Vários fatores podem existir para tornar um código *thread-unsafe*
    - Por exemplo, se o código mantém *estado* internamente, para salvar valores entre uma chamada e outra
    - Mudanças de estado podem causar problema quando dois threadas executam simultaneamente

Thread-Safety (cont.)

- Observação prática:
  - Várias funções da biblioteca C não são *thread-safe*
  - Por exemplo: *random()*
  - Algumas funções têm duas versões: *safe* e *unsafe*
    - func( ) (*unsafe*) e func_r( ) (*safe*)
  - Obviamente, programas com múltiplos threads não podem chamar rotinas que não sejam *thread-safe*
    - Erro não é incomum na prática!