



TÓPICOS ESPECIAIS – SISTEMAS EMBARCADOS

PROGRAMAÇÃO PARALELA E TEMPO REAL

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

Um pouco de paralelismo com outras libs

Exemplo em Java, com Threads, Locks, Synchronized, Runnable...

https://github.com/josenalde/parallel_programming_rtos/blob/main/src/Count3Sync.java



API com maior abstração que threads POSIX, desenvolvida no final dos anos 90

- ideia: incluir em códigos sequencias trechos identificáveis como passíveis de paralelização
- mais simples, mas não tão flexível quanto o controle por Pthreads
- definições de regiões paralelas com diretiva `#pragma omp parallel` com suas várias opções
 - laços paralelos (`parallel for`)
 - regiões críticas (`atomic`, `critical`, `exclusive`)
 - escopo de variáveis (`shared`, `private`)
 - redução

Exemplos no github, de [hello world](#), [count3s](#), [redução](#), [sections](#)

Um pouco de paralelismo com outras libs

Problema proposto: multiplicação matriz vetor (resolução de sistemas lineares etc.)



a_{00}	a_{01}	\cdots	$a_{0,n-1}$
a_{10}	a_{11}	\cdots	$a_{1,n-1}$
\vdots	\vdots		\vdots
a_{i0}	a_{i1}	\cdots	$a_{i,n-1}$
\vdots	\vdots		\vdots
$a_{m-1,0}$	$a_{m-1,1}$	\cdots	$a_{m-1,n-1}$

x_0
x_1
\vdots
x_{n-1}

 $=$

y_0
y_1
\vdots
$y_i = a_{i0}x_0 + a_{i1}x_1 + \cdots a_{i,n-1}x_{n-1}$
\vdots
y_{m-1}

$$\begin{bmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} =$$

$$Ax = y$$

$$y_i = \sum_{j=0}^{n-1} a_{ij}x_j$$

Pseudocódigo serial:

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

E para paralelizar? Ideias?

Um pouco de paralelismo com outras libs

Supondo $m=n=6$ com 3 threads...

OpenMP®

Thread	Components of y
0	$y[0], y[1]$
1	$y[2], y[3]$
2	$y[4], y[5]$

a_{00}	a_{01}	\cdots	$a_{0,n-1}$
a_{10}	a_{11}	\cdots	$a_{1,n-1}$
\vdots	\vdots		\vdots
a_{i0}	a_{i1}	\cdots	$a_{i,n-1}$
\vdots	\vdots		\vdots
$a_{m-1,0}$	$a_{m-1,1}$	\cdots	$a_{m-1,n-1}$

x_0	y_0
x_1	y_1
\vdots	\vdots
\vdots	$y_i = a_{i0}x_0 + a_{i1}x_1 + \cdots a_{i,n-1}x_{n-1}$
\vdots	\vdots
x_{n-1}	y_{m-1}

Thread 0, para calcular $y[0]$ faria como abaixo, acessando todos os elementos da linha 0 e do vetor x :

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

Mais genericamente, a thread responsável por calcular $y[i]$ faria:

```
y[i] = 0.0;  
for (j = 0; j < n; j++)  
    y[i] += A[i][j] * x[j]; //vetor x totalmente compartilhado
```

Um pouco de paralelismo com outras libs



Supondo $m=n=6$ com 3 threads...

Thread	Components of y
0	$y[0], y[1]$
1	$y[2], y[3]$
2	$y[4], y[5]$

Considerando que m é divisível por t , cada thread q irá manipular

m/t elementos

Primeiro: $q \times m/t$

Último: $(q+1) \times m/t - 1$

$m/t = 2$

Thread 0

Primeiro: 0

Último: $(0+1) \times 2 - 1 = 1$

Thread 1

Primeiro: $1 \times 2 = 2$

Último: $(1+1) \times 2 - 1 = 3$

Thread 2:

Primeiro: $2 \times 2 = 4$

Último: $(2 + 1) \times 2 - 1 = 5$

```
void matVectParallel(void *tid) {  
    long my_tid = (long) tid;  
    int i, j;  
    int local_m = m/thread_count;  
    int my_first_row = my_tid * local_m;  
    int my_last_row = (my_tid+1)*local_m - 1;  
    for (i = my_first_row; i <= my_last_row; i++) {  
        y[i] = 0.0;  
        for (j = 0; j < n; j++)  
            y[i] += A[i][j] * x[j];  
    }  
    return NULL;  
}
```

// com a matriz A, vetor x, vetor y, m e n globais

Exercício: transcrever para openMP com reduction