Computação Concorrente (DCC/UFRJ)

Aula 2: Projeto e avaliação de uma aplicação concorrente

Prof. Silvana Rossetto

13 de agosto de 2019

Problema inicial

Projete um algoritmo concorrente para implementar uma multiplicação matriz-vetor na forma $A * X = B$, sendo A uma matriz de dimensão n x m, X um vetor de dimensão m e B um vetor de dimensão n

$b_0 = a_{0,0} * x_0 + a_{0,1} * x_1 + ... + a_{0,n-1} * x_{n-1}$

...

$b_i = a_{i,0} * x_0 + a_{i,1} * x_1 + ... + a_{i,n-1} * x_{n-1}$

Primeira alternativa

Número variável de threads, cada thread calcula **um elemento** do vetor de saida:

```
void *calcula(void * tid) {
   int id = * (int *) tid;
   int i;
   b[id] = 0;
   for(i=0; i<M; i++) {
      b[id] += a[id][i] * x[i];
   }
}
```

Segunda alternativa

Número fixo de threads, cada thread calcula **um subconjunto** do vetor de saída:

```
void *calcula(void * tid) {
   int id = * (int *) tid;
   int i;
   for( ; id<N; id+=NTHREADS) {
      b[id] = 0;
      for(i=0; i<M; i++) {
         b[id] += a[id][i] * x[i];
      }
   }
}
```

Multiplicação de matrizes

Implementar uma versão concorrente para o problema de **multiplicação de matrizes** na forma $A * B = C$, sendo A, B e C matrizes quadradas de dimensão $N \times N$

Algoritmo sequencial para multiplicação de matrizes

```
//Aloca memória para as matrizes
...
//Carrega as matrizes de entrada
...
//Processa a multiplicação
...
//Exibe a matriz resultante
...
//Libera o espaço de memória alocada
...
```

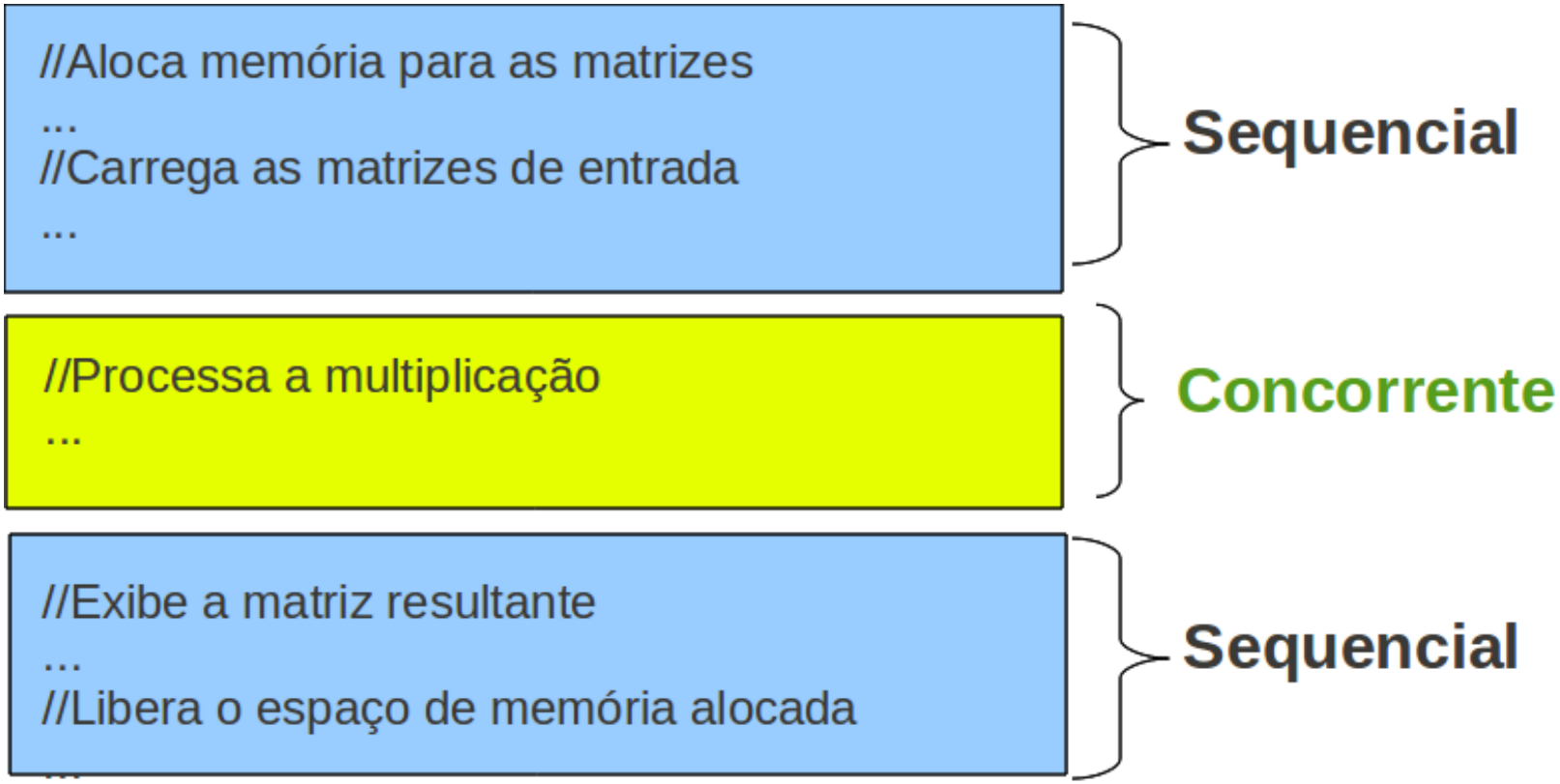
Passos para o projeto de uma solução concorrente

..a partir de uma solução sequencial

1. PASSO 1: análise do algoritmo sequencial
2. PASSO 2: cálculo do ganho de desempenho previsto
3. PASSO 3: projeto e implementação do algoritmo concorrente
4. PASSO 4: avaliação dos resultados

PASSO 1: análise do algoritmo básico

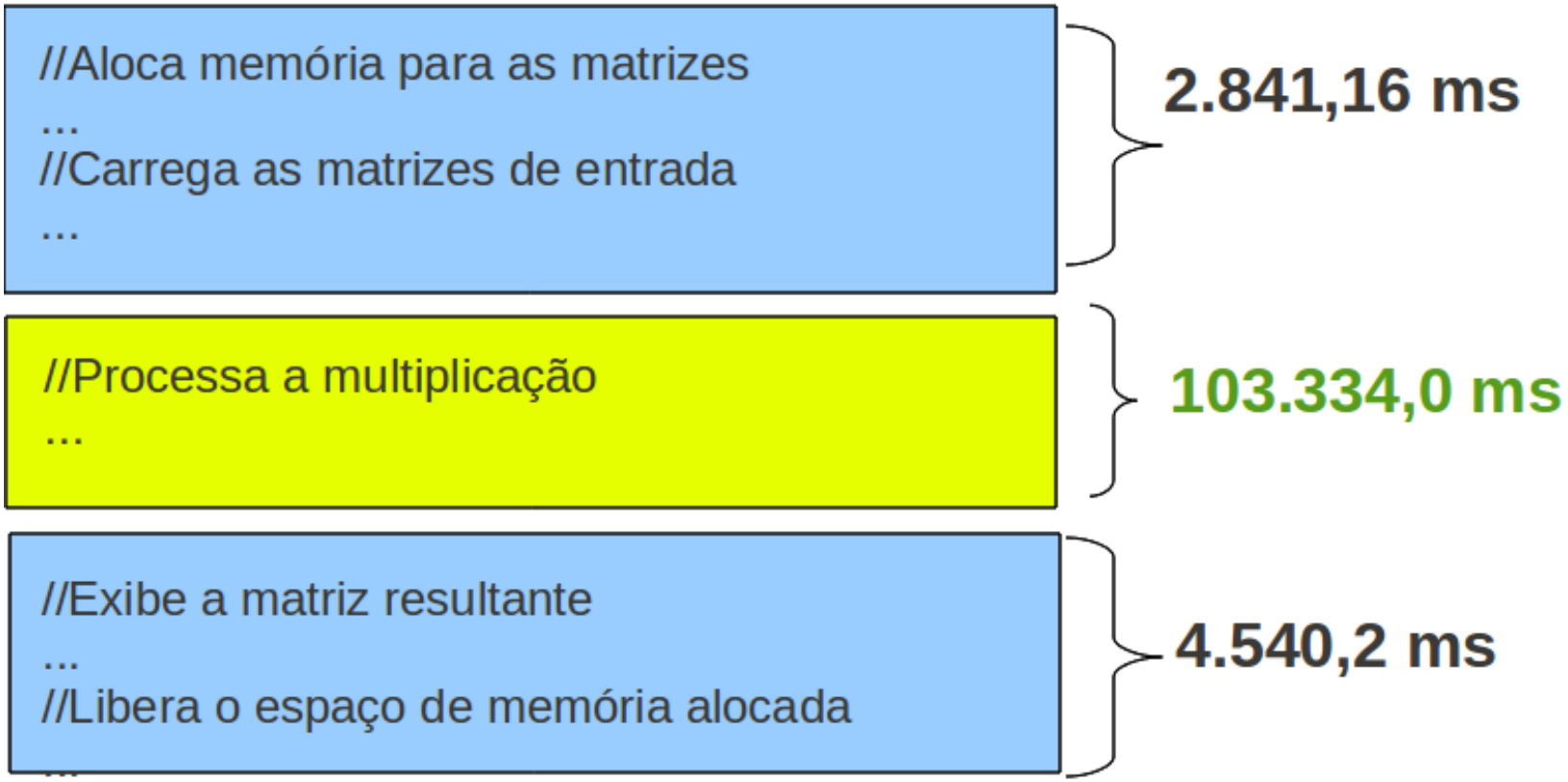
Quais são as etapas principais do programa?



PASSO 1: análise do algoritmo básico

Qual é o tempo de processamento de cada uma das etapas principais do programa?

Matrizes de entrada: A2048x2048.txt



PASSO 2: cálculo do ganho previsto

Lei de Amdahl

- Estima o ganho de velocidade de execução de um programa usando vários processadores
- **O ganho de velocidade da execução é dado por:** $T(sequencial)/T(concorrente)$

Lei de Amdahl para estimar ganho da concorrência

Tempo sequencial

- t_s: tempo da parte sequencial do programa (que não será dividida entre threads)
- $t_p(1)$: tempo da parte concorrente do programa usando um processador
- $T(1)$: tempo total do programa usando um processador $(t_s + t_p(1))$

Tempo concorrente

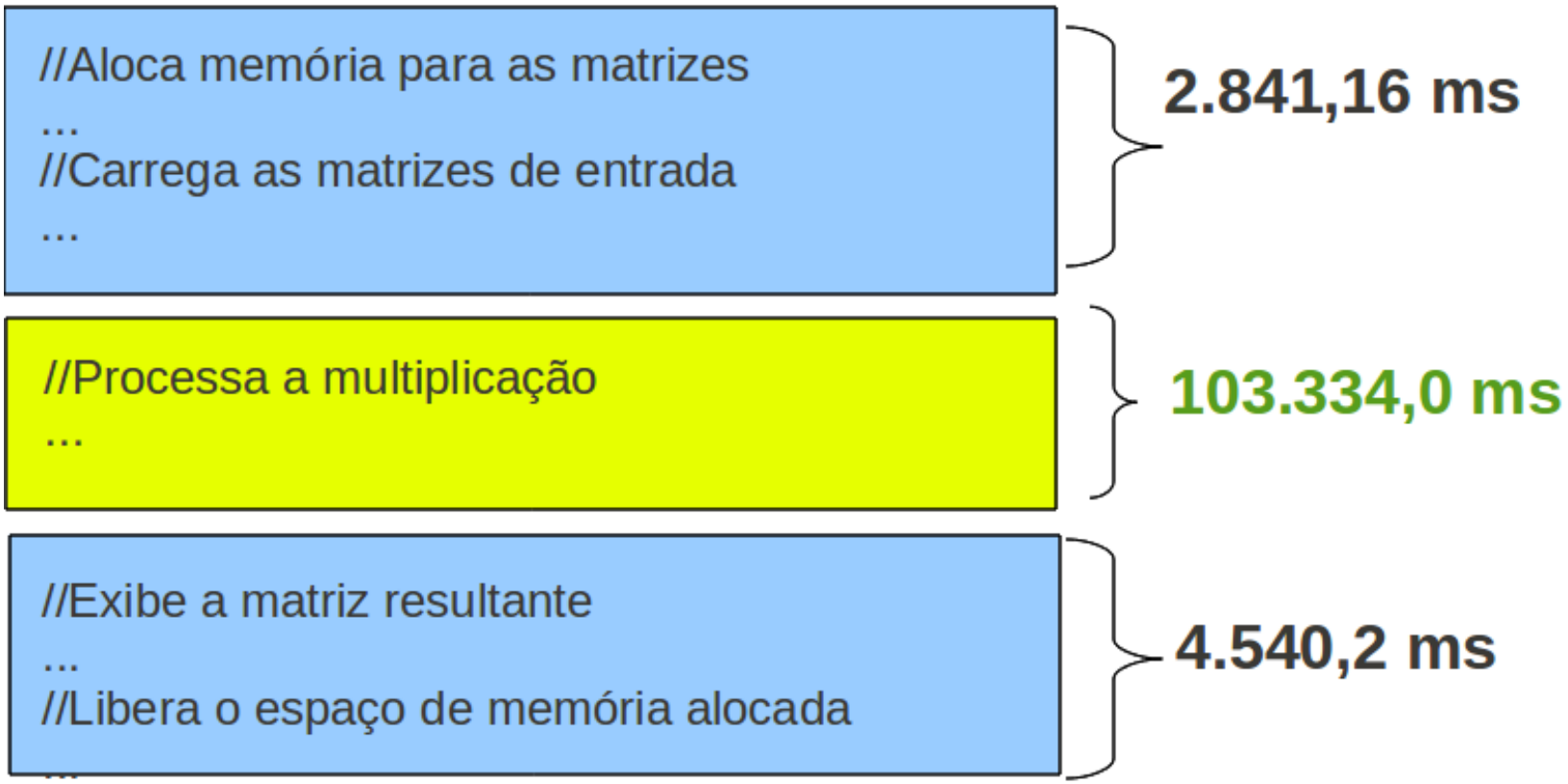
- t_s: tempo da parte sequencial do programa (que não será dividida entre threads)
- $t_p(P)$: tempo da parte paralela do programa usando P processadores $(t_p(1)/P)$
- $T(P)$: tempo total do programa usando P processadores $(t_s + t_p(P))$.

Lei de Amdahl para estimar ganho da concorrência

O ganho de velocidade da execução é dado por: $T(1)/T(P)$

Voltando ao problema de multiplicação de matrizes...

Matrizes de entrada: A2048x2048.txt



Qual será o ganho estimado de execução em uma máquina com 2 processadores?

Problema de multiplicação de matrizes

- O tempo total sequencial (T_s) é igual a **110.715,36**
- Com 2-processadores, a parte paralela gastará $p/n = 103334/2 =$ **51.667,0**
- A parte sequencial (entrada + saída) continuará igual a **7.381,36**
- O ganho estimado será $S = \frac{110715.36}{(7381.36+51667)} = 1{,}87$

Exercício

Considere um programa com 10 atividades (tempo similar), das quais 8 podem executar em paralelo (ao mesmo tempo). Qual será seu ganho de execução em uma máquina com 4 processadores?

Exercício

Considere um programa com 10 atividades (tempo similar), das quais 8 podem executar em paralelo (ao mesmo tempo). Qual será seu ganho de execução em uma máquina com 4 processadores?

Resposta

- seja **p** a fração da tarefa que pode ser executada em paralelo
- assumindo que o tempo para um processador completar a aplicação seja de 1 unidade de tempo, com n-processadores a parte paralela gastará p/n e a parte sequencial $1-p$, somando $(1-p+p/n)$, então, $S = \frac{1}{(1-p+p/n)}$
- $p = 8/10 = 4/5$, então $1-p = 1-4/5 = 1/5$
- daí, $S = \frac{1}{(1/5+(4/5)/4)} = \frac{1}{(2/5)} =$**2,5**

PASSO 3: projeto do algoritmo concorrente

Alternativa escolhida para divisão das tarefas entre as threads:

- Cada thread calcula um subconjunto de linhas (alternadas) da matriz de saída
- O número de threads é determinado pelo número de processadores da máquina, independente da dimensão das matrizes
- Não é necessário que o número de linhas seja divisível pelo número de threads

PASSO 3: implementação do algoritmo concorrente

Função principal

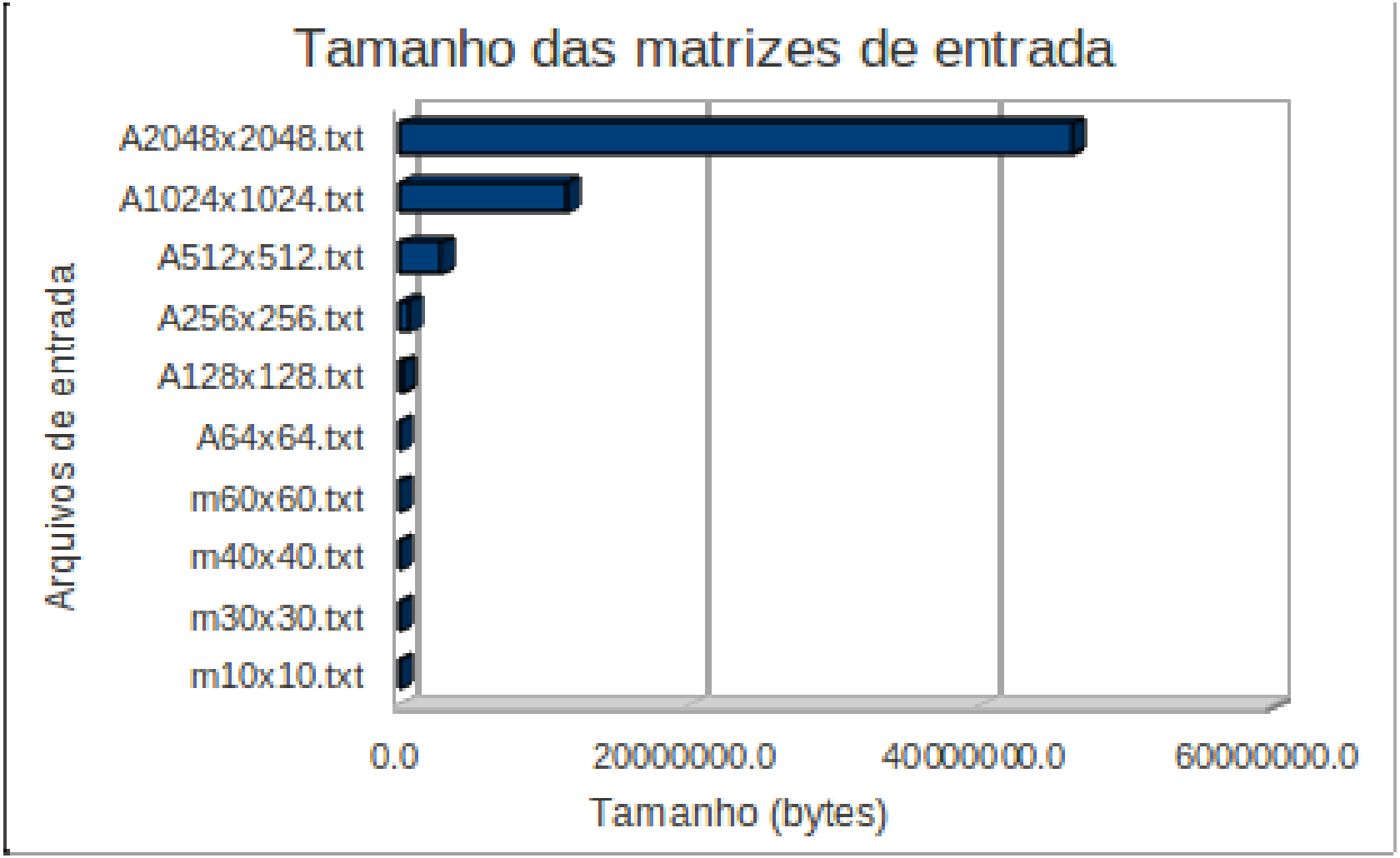
```
// matrizes quadradas de dimensao NxN
// nthreads: variavel global (numero de threads)
pthread_t *tid;
tid = (pthread_t*) malloc(sizeof(pthread_t)*nthreads);
int *arg;
//cria as threads
for (k=0; k<nthreads; k++) {
 if(arg = malloc(sizeof(int))) *arg=k; //...else EXIT
 pthread_create(&tid[k],NULL,calculaLinha,(void*)arg)
}
//espera todas as threads terminarem
for (k=0; k<nthreads; k++)
 pthread_join(tid[k], NULL)) {
...
```

PASSO 3: implementação do algoritmo concorrente

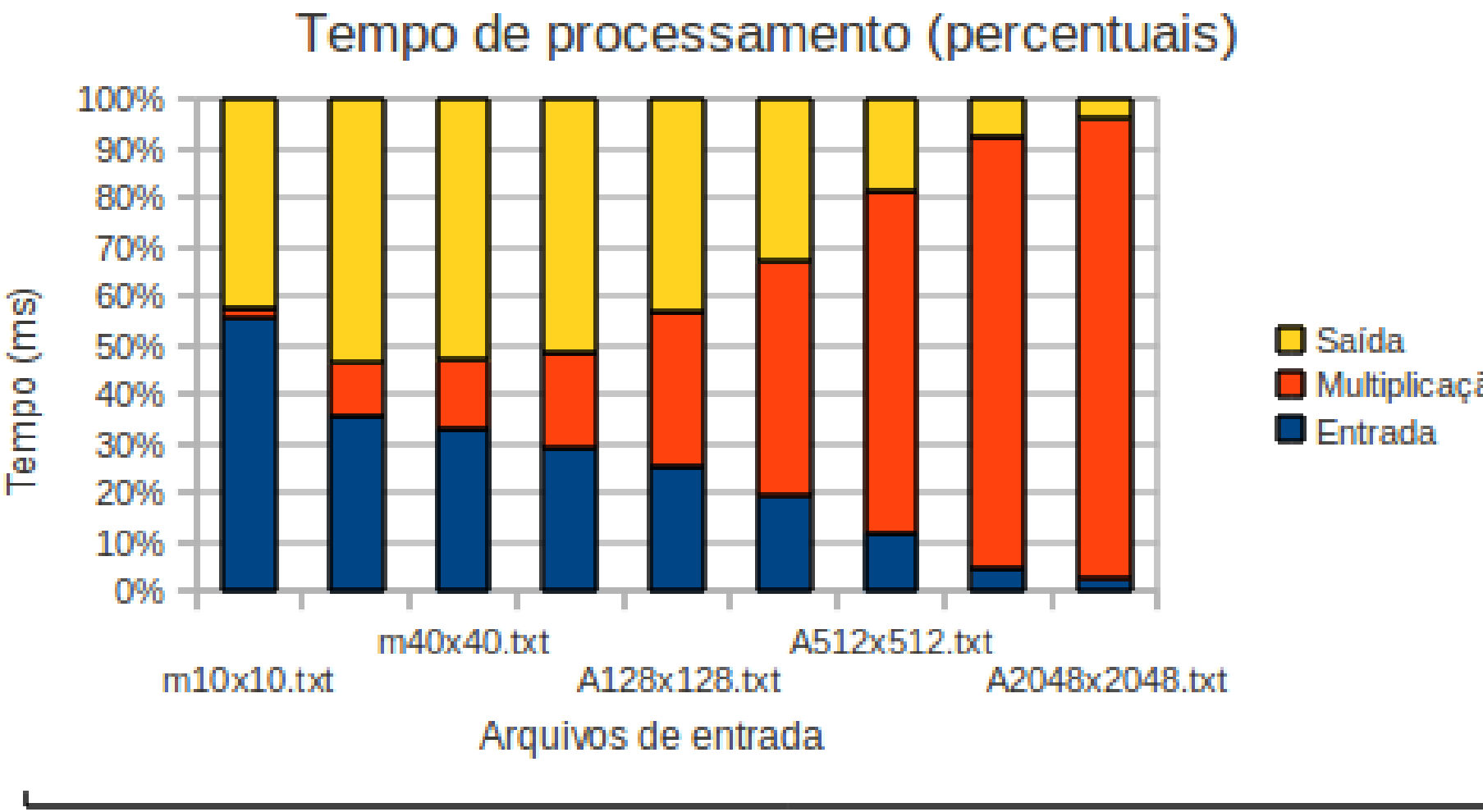
Função atribuída às threads

```
void *calculaLinha(void *tid){
  int linha = * (int*) tid;
  int k, coluna; float soma;
  for( ; linha<N; linha+=nthreads) {
     for(coluna=0; coluna<N; coluna++){
        soma = 0;
        for(k=0; k<N; k++)
              soma += mat1[linha][k] * mat2[k][coluna];
        mat3[linha][coluna] = soma;
     }
  }
}
```

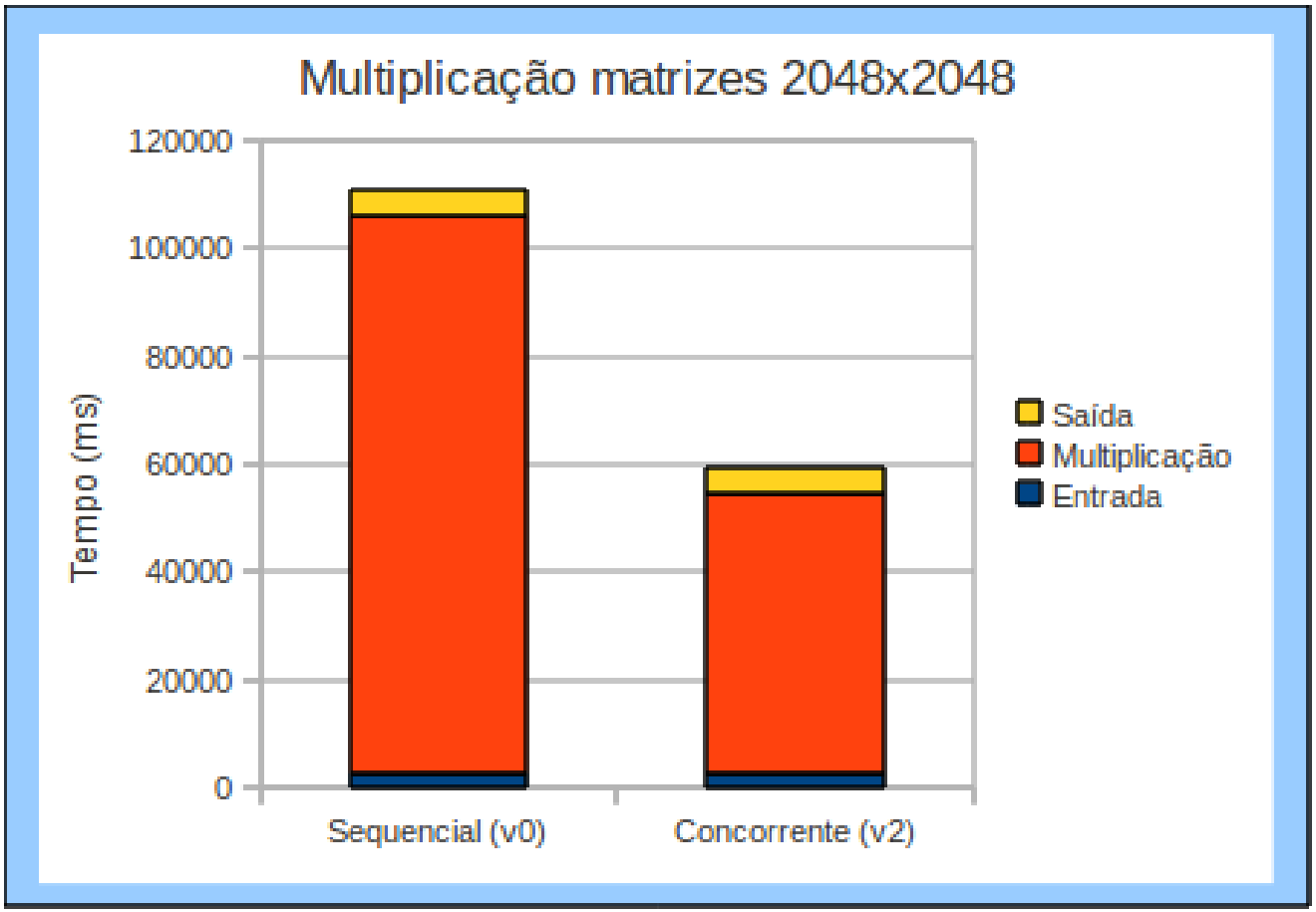
PASSO 4: Avaliação do ganho de desempenho



PASSO 4: Avaliação do ganho de desempenho



PASSO 4: Avaliação do ganho de desempenho



Ganho real: 110715.36/59283.36 = **1,867**

Custos associados à programação concorrente

Parâmetros do sistema ou biblioteca de thread

1. qual é o custo de criar uma thread?
2. qual é o custo da troca de contexto entre threads?
3. qual é o custo das operações de sincronização (*lock/wait*)?
4. qual é a demanda pelo uso da memória?

A aplicação concorrente pode gastar mais tempo de execução do que a aplicação sequencial dependendo do projeto da solução, do conjunto de dados de entrada, do ambiente computacional disponível

Exercícios

Considere um programa sequencial onde 1/4 do tempo de execução é gasto para alocação de memória e entrada de dados, 1/2 é gasto para o processamento dos dados e 1/4 é gasto para a impressão dos resultados finais. Se conseguirmos paralelizar a parte de processamento dos dados:

1. qual deve ser o número mínimo de processadores para reduzirmos 1/4 do tempo total de processamento da aplicação?
2. usando 8 processadores, qual será a fração de tempo máxima que conseguiremos reduzir do tempo total de execução?

Solução (1)

1. Seja T_{total} o tempo total de processamento da aplicação. Então $T_{total} = 1/4 + 1/2 + 1/4 = 1$. Seja $T_{desejado}$ o tempo total de processamento da aplicação desejado, que equivale a $3/4$ de T_{total}, e N o número de processadores usados. Paralelizando a parte de processamento de dados, teremos: $T_{paralelo-total} = 2/4 + ((1/2)/N)$, uma vez que a alocação de memória e a entrada e saída de dados continua sequencial. Então teremos: $T_{desejado} = T_{paralelo-total}$, daí: $3/4 = 2/4 + ((1/2)/N)$, então $1/4 = 1/2 * N$, dai: $2 * N = 4$, então $N = 2$.

Solução (2)

1. ...
2. Usando 8 processadores, dividiremos o tempo de processamento dos dados por 8. Então teremos: $T_{paralelo-total} = 2/4 + ((1/2)/8) = 9/16$. Portanto reduziríamos no máximo o tempo total de processamento da aplicação de 7/16.

Exercício

Considere o problema de encontrar **o maior elemento de um vetor não-ordenado**. Proponha um algoritmo concorrente para resolver esse problema considerando um vetor de tamanho N.

Referências bibliográficas

- https://computing.llnl.gov/tutorials/pthreads/