# Relatório do Exercício 4: Simulação de Linha de Produção com OpenMP

**Nome:** Daniel Araujo Cavassani (COMP 25)

**Data:** 22/11/2024

O objetivo do exercício foi implementar e paralelizar uma simulação de linha de produção com nEsta estágios e nProd produtos, utilizando OpenMP. A tarefa consistiu em modificar o programa fornecido para explorar o paralelismo, analisar os resultados com diferentes números de threads e correlacionar a simulação com um cenário real.

## Descrição da Paralelização

Para paralelizar o código, foi utilizado o OpenMP para simular os estágios operando em paralelo, respeitando as dependências entre os produtos. O trecho modificado do código principal está apresentado abaixo:

```
 // fabrica todos os produtos
#pragma omp parallel
 {
#pragma omp single
   {
     for (int prod = 1; prod <= nProd; prod++) {
       for (int esta = 1; esta <= nEsta; esta++) {
         // cria uma tarefa para cada estágio de cada produto
#pragma omp task depend(in: dummy[prod][esta - 1], dummy[prod - 1][esta])
depend(out: dummy[prod][esta])
         {
           // processamento do estágio
           Estagio(esta, prod, tIni);

           // imprime o tempo de fabricação quando o último estágio
terminar
           if (esta == nEsta) {
```

```
            printf("produto %2.0d fabricado apos %2.0lf segundos com %d
threads\n",
                    prod, WallTime() - tIni, nThreads);
          }
        }
      }
    }
  }
```

Cada estágio de um produto depende do término do estágio anterior do mesmo produto e do mesmo estágio do produto anterior.

As diretivas `#pragma omp task` com `depend` garantem a sincronização correta entre os estágios e produtos.

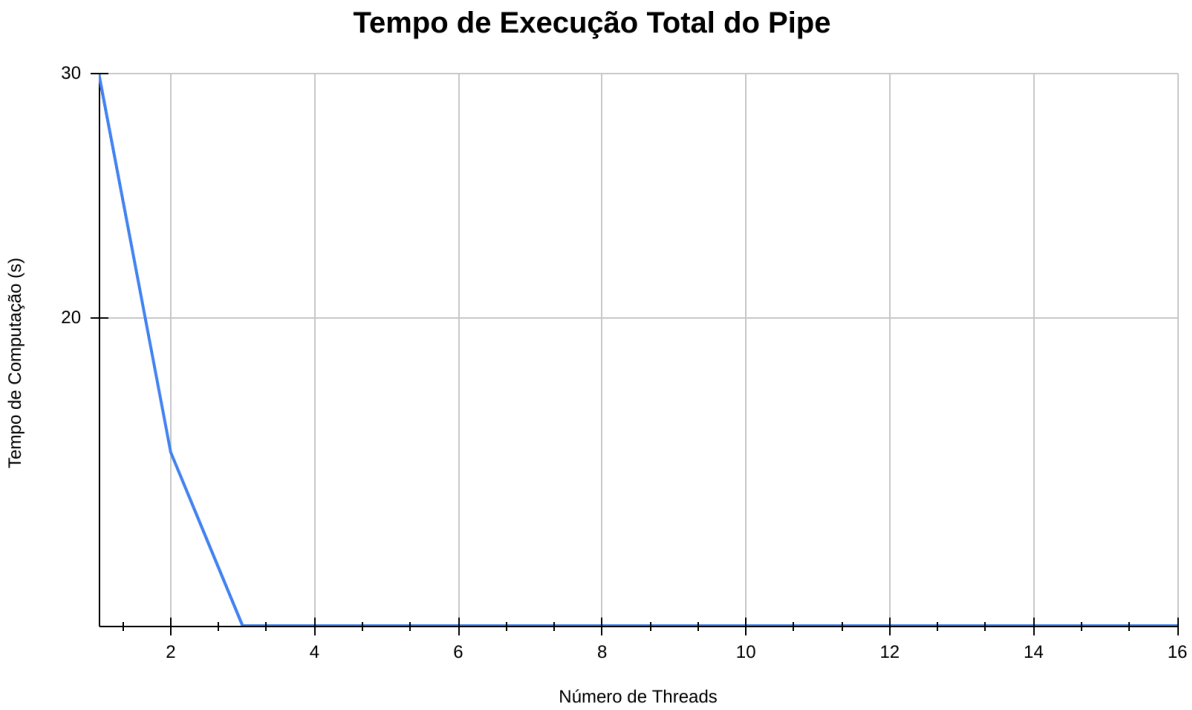
A diretiva `#pragma omp taskwait` assegura que o processamento de um produto esteja completo antes de imprimir seu tempo.

## Resultados

Os tempos de fabricação para nEsta=3n e nProd=10 foram medidos com diferentes números de threads. A tabela abaixo resume os tempos finais de execução:

| Número de Threads | |
|---|---|
| 1 | 30 |
| 2 | 16 |
| 3 | 12 |
| 4 | 12 |
| 8 | 12 |
| 16 | 12 |

Este gráfico mostra a redução no tempo de execução à medida que o número de processos aumenta:




## Cálculo do Speed-up

O speed-up foi calculado dividindo o tempo de execução com 1 processo pelo tempo de execução com múltiplos processos. Os resultados foram:

| Número de Threads | |
|---|---|
| 1 | 1.00 |
| 2 | 1.88 |
| 3 | 2.50 |
| 4 | 2.50 |
| 8 | 2.50 |
| 16 | 2.50 |

## Análise

Os resultados obtidos mostram claramente o comportamento da simulação de uma linha de produção paralelizada. Para nEsta=3 estágios e nProd=10 produtos:

1. **Comportamento com T=1 thread:**
    - Cada produto é processado sequencialmente, resultando no maior tempo total (30 segundos). Isso ocorre porque o processamento de cada estágio depende completamente do término do estágio anterior.
2. **Comportamento com T=2 threads:**
    - A introdução de paralelismo reduz o tempo total para 16 segundos. No entanto, não é possível atingir o tempo mínimo teórico, pois há mais estágios (nEsta=3) do que threads, resultando em uma certa sobreposição de execução.
3. **Comportamento com T≥3 threads:**
    - A partir de 3 threads, o tempo total estabiliza em 12 segundos, o limite teórico mínimo. Cada estágio pode ser processado simultaneamente, e não há mais dependências que impeçam o avanço.
4. **Threads adicionais (T>3):**
    - O uso de mais threads não melhora o desempenho, pois o número de estágios limita o paralelismo efetivo. Assim, threads extras ficam ociosas.

Esses resultados reforçam a importância de equilibrar o número de threads disponíveis com as demandas de paralelismo de uma aplicação. Em uma linha de produção real, cada thread pode ser vista como representando um operador ou unidade de trabalho, e o número ideal de threads equivale ao número de estágios no pipeline.

## Conclusão

A paralelização da simulação da linha de produção com OpenMP demonstrou resultados consistentes com os limites teóricos. O tempo de fabricação mínimo foi alcançado com 3 threads, representando o número de estágios da linha. Isso evidencia que o paralelismo é limitado pelas dependências intrínsecas do problema e que o uso excessivo de threads não traz benefícios adicionais.

O exercício destacou a relação direta entre o número de threads e os recursos físicos disponíveis, como operadores ou máquinas, em sistemas reais. Essa compreensão é crucial para projetar e otimizar pipelines paralelos, garantindo o uso eficiente de recursos e evitando sobrecarga ou desperdício de capacidade computacional.