Insper

SuperComputação

Aula 12 – Introdução a OpenMP

2020 – Engenharia

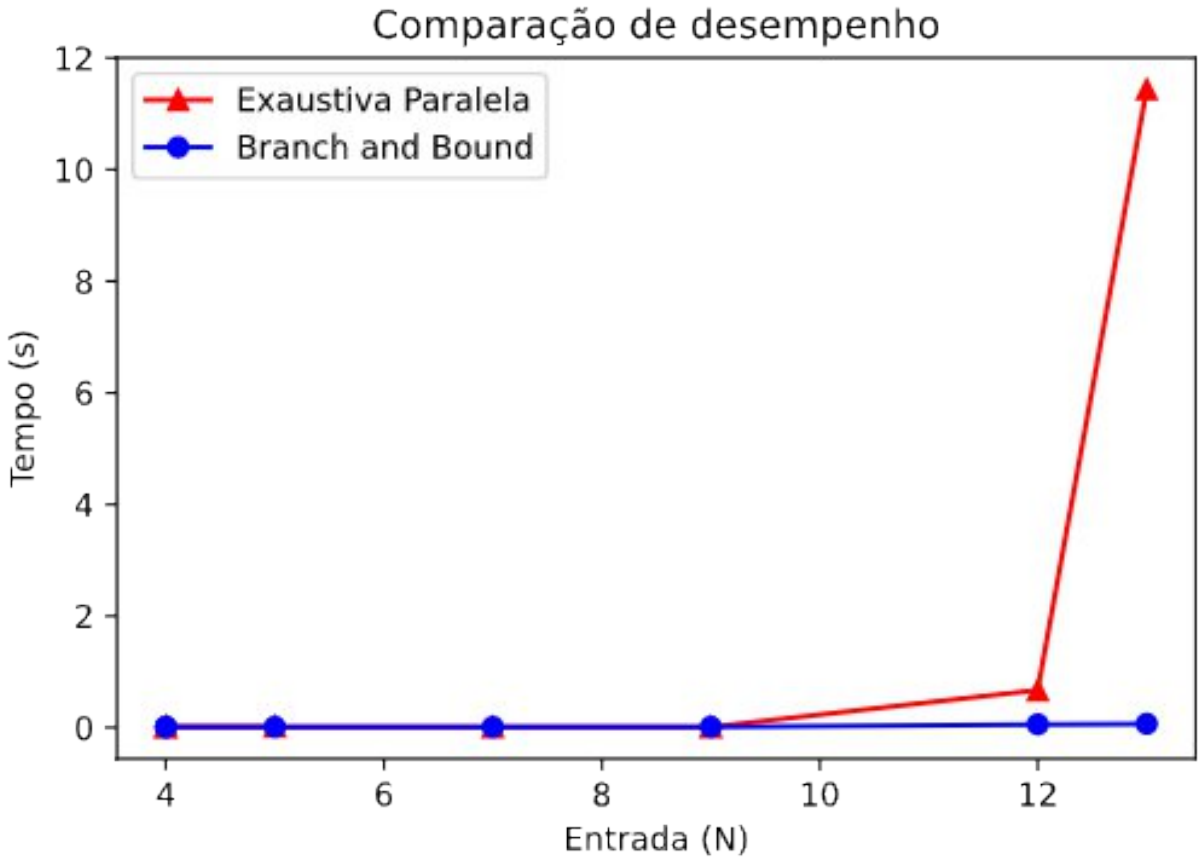
Luciano Soares <lpsoares@insper.edu.br>
Igor Montagner <igorsm1@insper.edu.br>

Solução de alto desempenho

1. **Algoritmos eficientes**
2. Implementação eficiente
    - Cache, paralelismo de instrução
    - Linguagem de programação adequada
3. **Paralelismo**

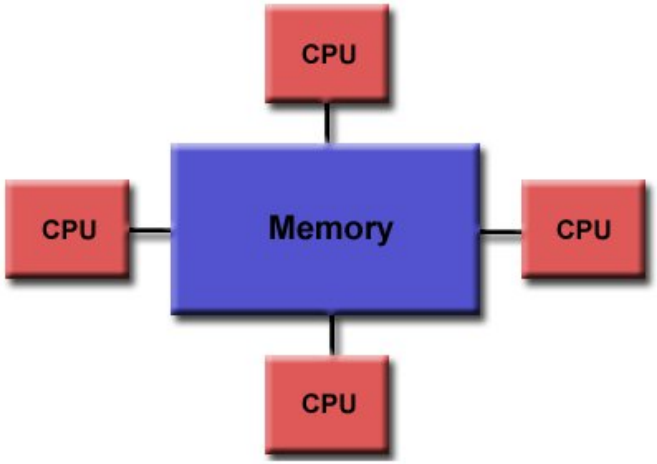
Discussão: o quanto algoritmos bons ajudam?

Algoritmos fazem a diferença! (II)

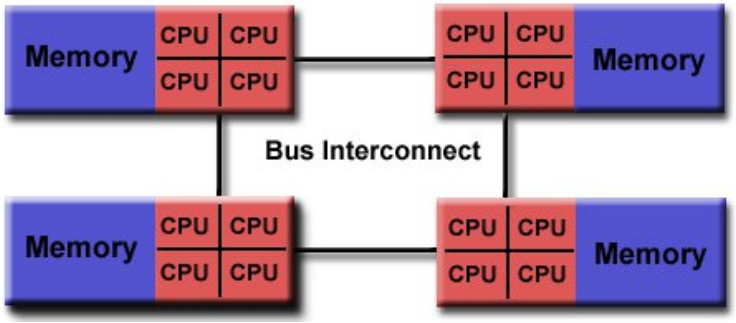


Insper

Sistemas Multi-core



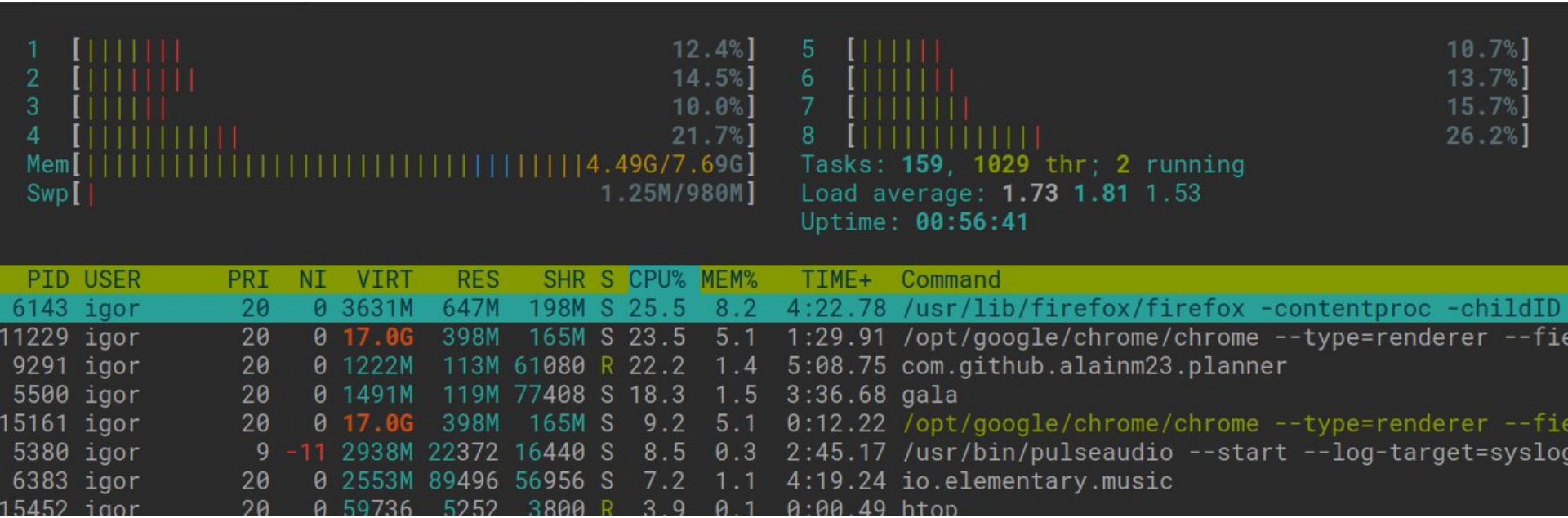
Uniform Memory Access



Non-Uniform Memory Access

Insper

Sistemas multi-core



Insper

Conceito 1: Dependência

Um loop tem uma **dependência** de dados sua execução correta depende da ordem de sua execução.

Isto ocorre quando **uma iteração depende de resultados calculados em iterações** anteriores.

Quando não existe nenhuma dependência em um loop ele é dito **ingenuamente paralelizável**.

Conceito 2: Paralelismo

Paralelismo de dados: faço em paralelo a mesma operação (lenta) para todos os elementos em um conjunto de dados (grande).

Paralelismo de tarefas: faço em paralelo duas (ou mais) tarefas independentes. Se houver dependências quebro em partes independentes e rodo em ordem.

Paralelismo Multi-core

Threads:

- Compartilham memória
- Sincronização de acessos

Processos:

- Troca de mensagens
- Possível distribuir em vários nós

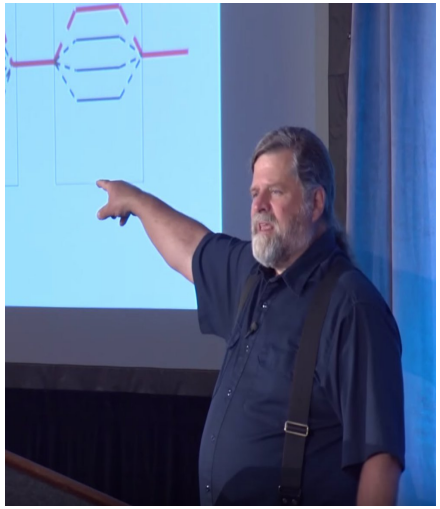
OpenMP

- Conjunto de extensões para C/C++ e Fortran
- Fornece construções que permitem paralelizar código em ambientes multi-core
- Padroniza práticas SMP + SIMD + Sistemas heterogêneos (GPU/FPGA)
- Idealmente funciona com mínimo de modificações no código sequencial

Fontes importantes

A brief Introduction to parallel programming

Tim Mattson
Intel Corp.
timothy.g.mattson@ intel.com



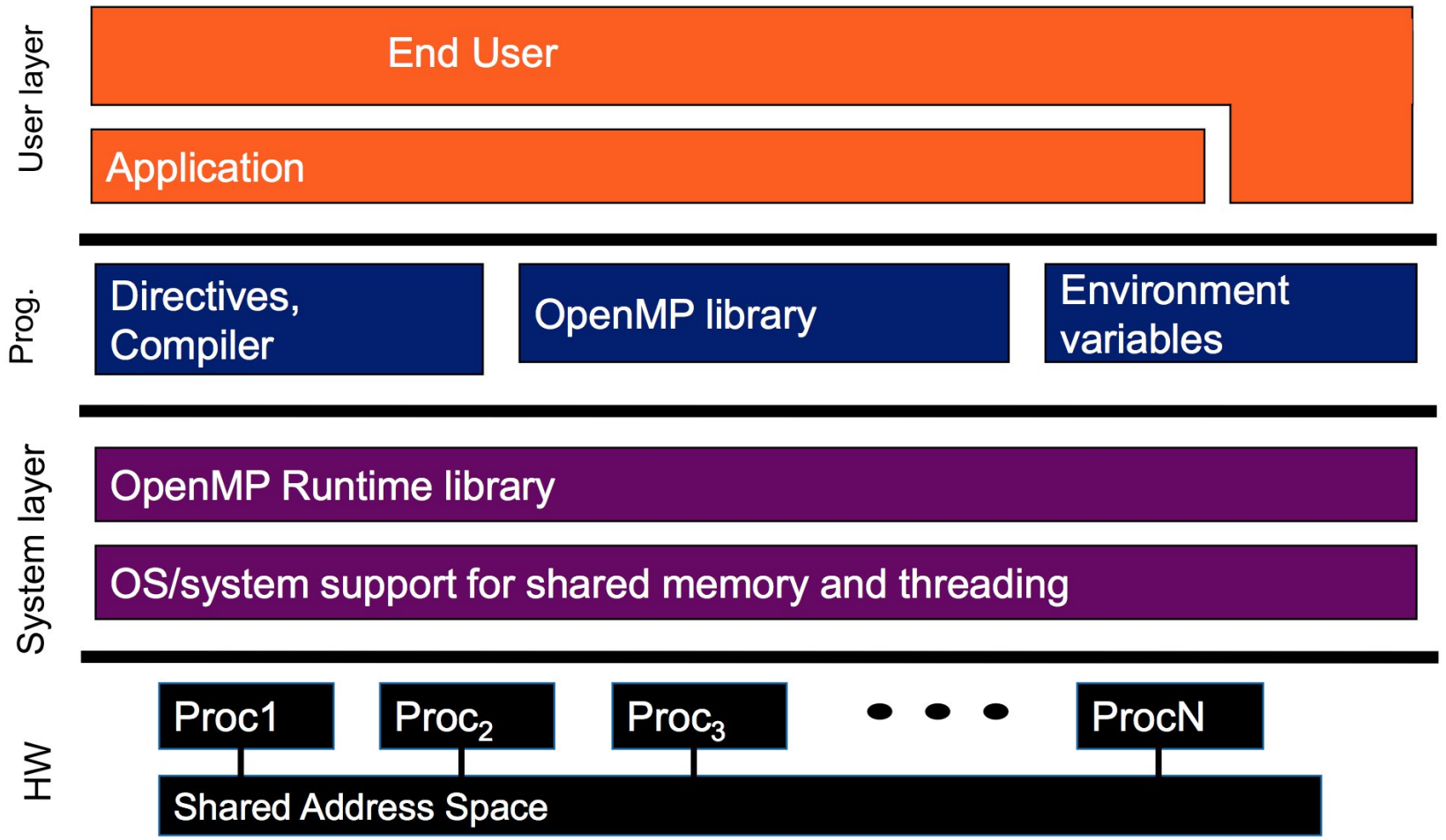
Vídeos:

https://www.youtube.com/watch?v=pRtTIW9-Nr0
https://www.youtube.com/watch?v=LRsQHDAqPHA
https://www.youtube.com/watch?v=dK4PITrQtjY
https://www.youtube.com/watch?v=WvoMpG_QvBU

Slides:

http://extremecomputingtraining.anl.gov/files/2016/08/Mattson_830aug3_HandsOnIntro.pdf

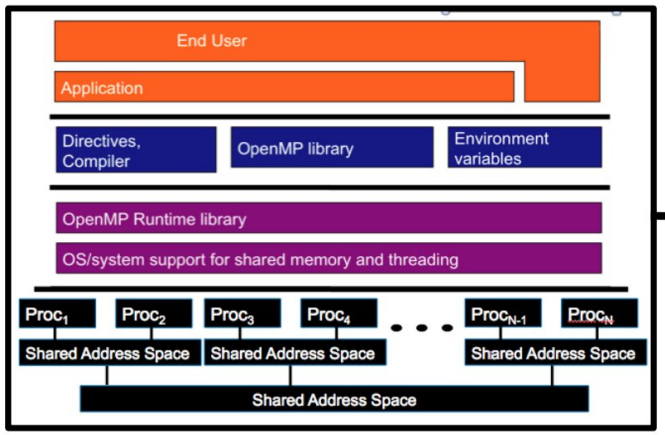
OpenMP (host / NUMA)



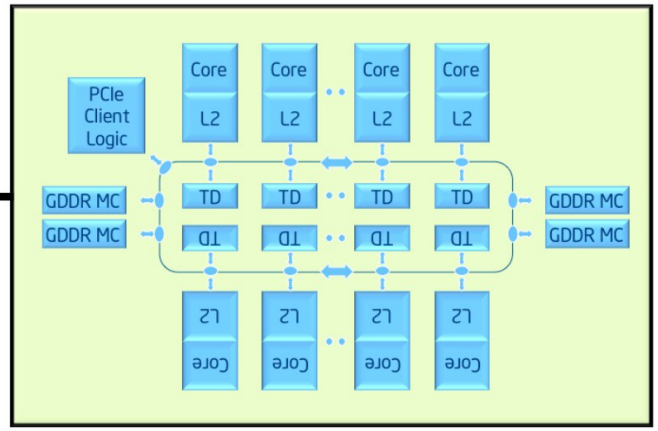
OpenMP (heterogêneo / target)

Version 4.0-4.5

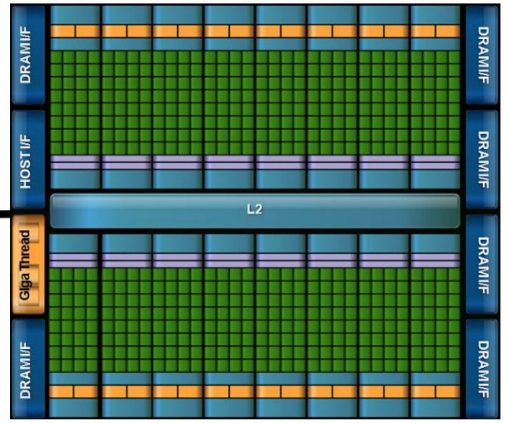
Supported (since OpenMP 4.0) with target, teams, distribute, and other constructs



Host



Target Device: Intel® Xeon Phi™ coprocessor



Target Device: GPU

Insper

OpenMP - sintaxe

Diretivas de compilação

```
#include <omp.h>
#pragma omp construct [params]
```

Aplicadas a um bloco de código

limitado diretamente por { }

for (…) { }

Com join implícito

Atividade prática

Primeiros passos com OpenMP (20 minutos)

1. Rodar programas simples com OpenMP
2. Entender a criação de threads com #pragma omp parallel

Conceito 2: Paralelismo

Paralelismo de dados: faço em paralelo a mesma operação (lenta) para todos os elementos em um conjunto de dados (grande).

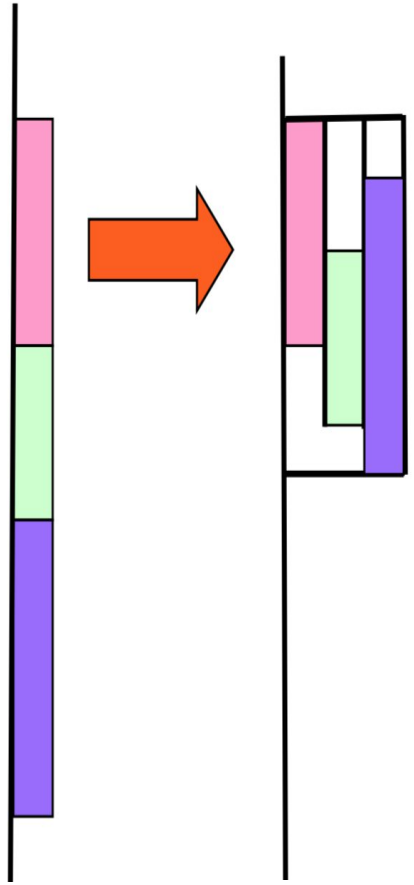
Paralelismo de tarefas: faço em paralelo duas (ou mais) tarefas independentes. Se houver dependências quebro em partes independentes e rodo em ordem.

O quê são tarefas?

A tarefa é definida em um bloco estruturado de código

Dentro de uma região paralela, a thread que encontra uma construção de tarefa irá empacotar todo o bloco de código e seus dados para execução

Tarefas podem ser aninhadas: isto é, uma tarefa pode gerar novas tarefas

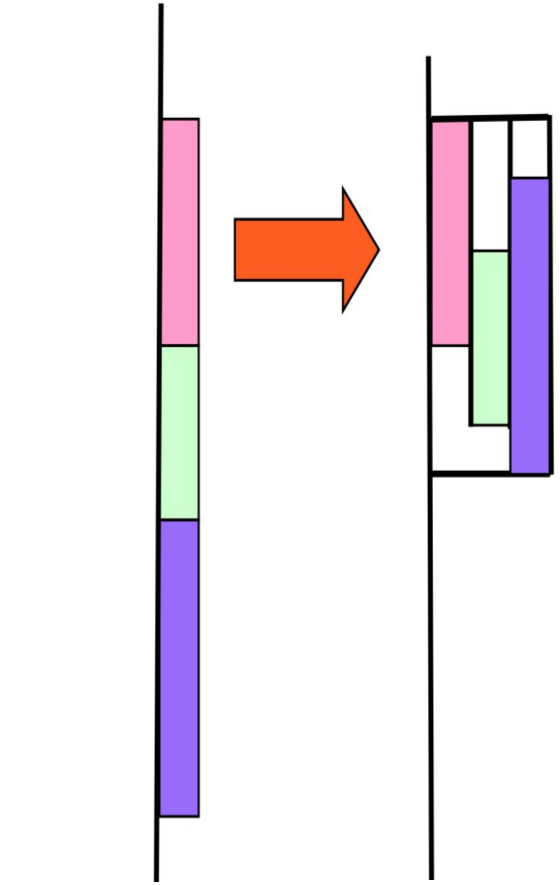


O quê são tarefas?

Cada thread pode ser alocada para rodar uma tarefa

Não existe ordenação no início das tarefas

Tarefa são unidades de trabalho independentes



Tarefas em OpenMP

#pragma omp task[clauses]

```
#pragma omp parallel
{
        #pragma omp master
        {
                #pragma omp task
                    func1();
                #pragma omp task
                    func2();
                #pragma omp task
                    func3();
        }
}
```

Crie um conjunto de threads

Thread 0 organiza as tarefas

Tarefas executadas por alguma thread em alguma ordem

Todas as tarefas devem ser concluídas antes que esta barreira seja liberada

Tarefas em OpenMP

#pragma omp task[clauses]

```
#pragma omp parallel
{
        #pragma omp master
        {
                #pragma omp task
                    func1();
                #pragma omp task
                    func2();
                #pragma omp taskwait
                #pragma omp task
                    func3();
        }
}
```

Task func3() só é criada depois de func1() e func2() acabarem.

Cuidado - Tarefas são caras!

- Muitas tarefas geradas podem deixam o código mais lento
- Ao gerar tarefa o sistema terá que suspender a execução por um tempo

```
#pragma omp master
{
        for(i=0;i<UMZILHAO;i++)
        #pragma omp task
                process(item[i]);
}
```

Atividade prática

Primeiros passos com Tarefas (20 minutos)

1. Entender a criação de tarefas com #pragma omp task
2. Rodar primeiras funções em paralelo.

Atividade prática

Problema prático (20 minutos)

1. Transformar um código sequencial em paralelo usando tarefas.

Insper

www.insper.edu.br