# UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

**Curso:** Engenharia de Software.

**Disciplina:** Sistemas Operacionais.

**Aluno:** Eduardo Caversan da Silva Rocha, 2475391

# Introdução e objetivo:

O exercício tem o objetivo de explorar a comparação de desempenho entre as abordagens de soma sequencial (laço for simples), OpenMP (biblioteca que permite paralelismo), TBB (biblioteca que permite paralelismo), incluindo o cálculo de média e desvio padrão dos tempos de execução de um algoritmo que soma os elementos de um vetor de tamanho 10^8 (100 milhões de posições).

# Metodologia:

Para o desenvolvimento, compilação e execução do código do experimento, o sistema operacional escolhido foi o Linux. A escolha foi baseada na preferência do aluno, tendo em vista que ele possui conhecimento básico nesse sistema operacional e julgou ser mais fácil de implementar o algoritmo no mesmo. É importante citar que durante o desenvolvimento do projeto, nenhuma IDE foi utilizada, apenas o bloco de notas que já vem na distro Linux utilizada.

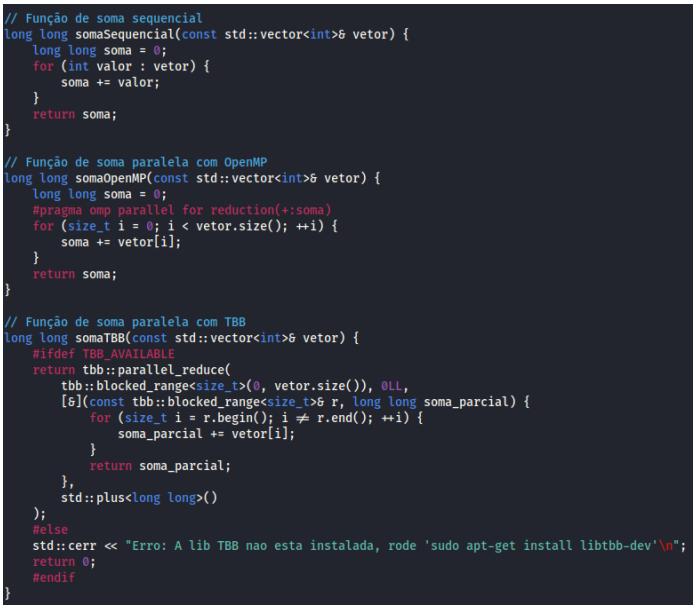
Para iniciar o desenvolvimento, foi necessário rodar dois comandos:

1. sudo apt-get update
2. sudo apt-get install libtbb-dev

Os comandos acima garantem que as dependências necessárias estejam instaladas para que o código possa ser compilado e implementado.

Após ter o ambiente configurado, o código foi desenvolvido para que em uma execução da função principal (main) ele execute 30 vezes a soma do vetor para cada tipo de método, imprima no console o tempo de cada uma das execuções e tempo médio e desvio padrão de cada um dos métodos.

## Imagem das funções principais do código:



```cpp
// Função de soma sequencial
long long somaSequencial(const std::vector<int>& vetor) {
    long long soma = 0;
    for (int valor : vetor) {
        soma += valor;
    }
    return soma;
}

// Função de soma paralela com OpenMP
long long somaOpenMP(const std::vector<int>& vetor) {
    long long soma = 0;
    #pragma omp parallel for reduction(+:soma)
    for (size_t i = 0; i < vetor.size(); ++i) {
        soma += vetor[i];
    }
    return soma;
}

// Função de soma paralela com TBB
long long somaTBB(const std::vector<int>& vetor) {
    #ifdef TBB_AVAILABLE
    return tbb::parallel_reduce(
        tbb::blocked_range<size_t>(0, vetor.size()), 0LL,
        [&](const tbb::blocked_range<size_t>& r, long long soma_parcial) {
            for (size_t i = r.begin(); i != r.end(); ++i) {
                soma_parcial += vetor[i];
            }
            return soma_parcial;
        },
        std::plus<long long>()
    );
    #else
    std::cerr << "Erro: A lib TBB nao esta instalada, rode 'sudo apt-get install libtbb-dev'\n";
    return 0;
    #endif
}
```

## Comando para compilar o código:

- g++ -std=c++11 -o somavetor -DTBB_AVAILABLE -fopenmp "caminho para o arquivo de código fonte .cpp” -ltbb (No caso deste experimento o aluno criou o arquivo de código fonte diretamente no diretório Desktop, logo o caminho ficou: Desktop/somavetor.cpp”).

## Comando para executar o código já compilado:

- ./somavetor (Esse passo é subjetivo, já que é o caminho para o arquivo de código compilado pode variar de acordo com o usuário, porém no caso do aluno, esse foi o caminho escolhido).

## Link para acesso completo ao código fonte:

- Repositório GitHub.

# Método sequencial

## Tabela de execuções:

| Execução | Tempo (s) |
|---|---|
| 1 | 0.182443 |
| 2 | 0.170775 |
| 3 | 0.169755 |
| 4 | 0.171179 |
| 5 | 0.171177 |
| 6 | 0.168999 |
| 7 | 0.171103 |
| 8 | 0.169119 |
| 9 | 0.170034 |
| 10 | 0.172191 |
| 11 | 0.171621 |
| 12 | 0.169279 |
| 13 | 0.171595 |
| 14 | 0.171594 |

| | |
|---|---|
| 15 | 0.173545 |
| 16 | 0.170251 |
| 17 | 0.172155 |
| 18 | 0.169468 |
| 19 | 0.170552 |
| 20 | 0.170324 |
| 21 | 0.168249 |
| 22 | 0.172274 |
| 23 | 0.173473 |
| 24 | 0.172324 |
| 25 | 0.171157 |
| 26 | 0.170353 |
| 27 | 0.169954 |
| 28 | 0.170239 |
| 29 | 0.170239 |
| 30 | 0.172124 |

## Estatísticas

- **Tempo Médio:** 0.17111 segundos.
- **Desvio Padrão:** 0.0024204 segundos.

# Método utilizando OpenMP

## Tabela de execuções:

| Execução | Tempo (s) |
|---|---|
| 1 | 0.0733653 |
| 2 | 0.10496 |
| 3 | 0.0778493 |
| 4 | 0.0662442 |
| 5 | 0.102948 |
| 6 | 0.0924513 |
| 7 | 0.090551 |
| 8 | 0.090567 |

| | |
|---|---|
| 9 | 0.083876 |
| 10 | 0.080788 |
| 11 | 0.155538 |
| 12 | 0.122219 |
| 13 | 0.116886 |
| 14 | 0.077657 |
| 15 | 0.0964234 |
| 16 | 0.0816548 |
| 17 | 0.0791244 |
| 18 | 0.117791 |
| 19 | 0.0826533 |
| 20 | 0.0864717 |
| 21 | 0.0884553 |
| 22 | 0.080988 |
| 23 | 0.0857642 |
| 24 | 0.083476 |
| 25 | 0.120759 |
| 26 | 0.108562 |
| 27 | 0.0912554 |
| 28 | 0.092384 |
| 29 | 0.0825424 |
| 30 | 0.0934438 |

## Estatísicas

- **Tempo Médio:** 0.10235 segundos.
- **Desvio Padrão:** 0.0223194 segundos.

# Método utilizando TBB

## Tabela de execuções:

| Execução | Tempo (s) |
|---|---|
| 1 | 0.144406 |
| 2 | 0.148461 |

| | |
|---|---|
| 3 | 0.101701 |
| 4 | 0.129795 |
| 5 | 0.151899 |
| 6 | 0.0932397 |
| 7 | 0.0905743 |
| 8 | 0.0885749 |
| 9 | 0.0906716 |
| 10 | 0.0777828 |
| 11 | 0.100345 |
| 12 | 0.0899597 |
| 13 | 0.0978575 |
| 14 | 0.0820044 |
| 15 | 0.101721 |
| 16 | 0.0939782 |
| 17 | 0.104097 |
| 18 | 0.083258 |
| 19 | 0.0906969 |
| 20 | 0.0824108 |
| 21 | 0.0894924 |
| 22 | 0.137417 |
| 23 | 0.0858938 |
| 24 | 0.0883686 |
| 25 | 0.0864508 |
| 26 | 0.113629 |
| 27 | 0.0947946 |
| 28 | 0.0745089 |
| 29 | 0.0751013 |
| 30 | 0.107213 |

## Estatísticas

- **Tempo médio:** 0.0998768 segundos.
- **Desvio padrão:** 0.0211929 segundos.

## Conclusão:

Como observado pelos dados fornecidos acima, o método com menor tempo médio e execução mais rápida foi utilizando paralelismo com TBB, enquanto o com maior tempo médio e execução mais lenta foi a soma sequencial (laço for simples).

O motivo pelo qual esse acontece é que o paralelismo permite que a tarefa “grande” seja dividida e processada em partes, fazendo com que se diminuam a quantidade de operações, e consequentemente o tempo de execução.

## Motivo pelo qual os gráficos não foram plotados e exibidos:

Devido às especificações fornecidas pelo professor no moodle, o relatório não poderia passar de 5 páginas, portanto caso os gráficos fossem adicionados aqui, não haveria espaço para as demais informações.