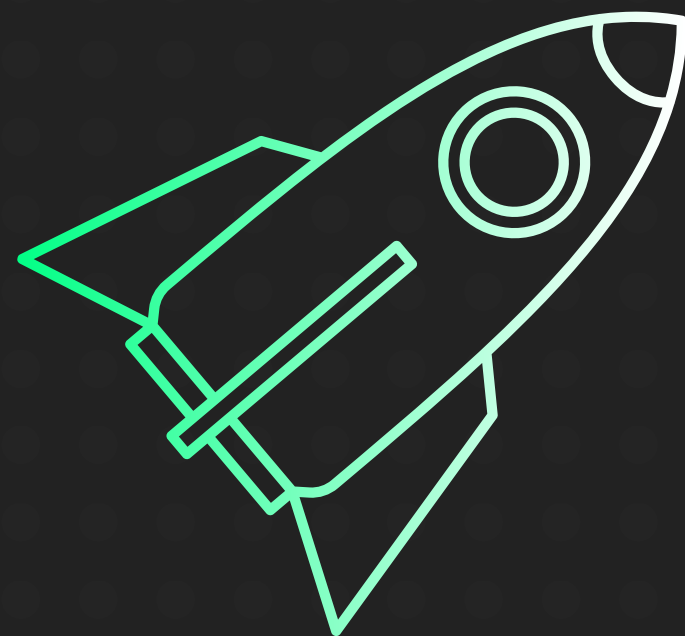
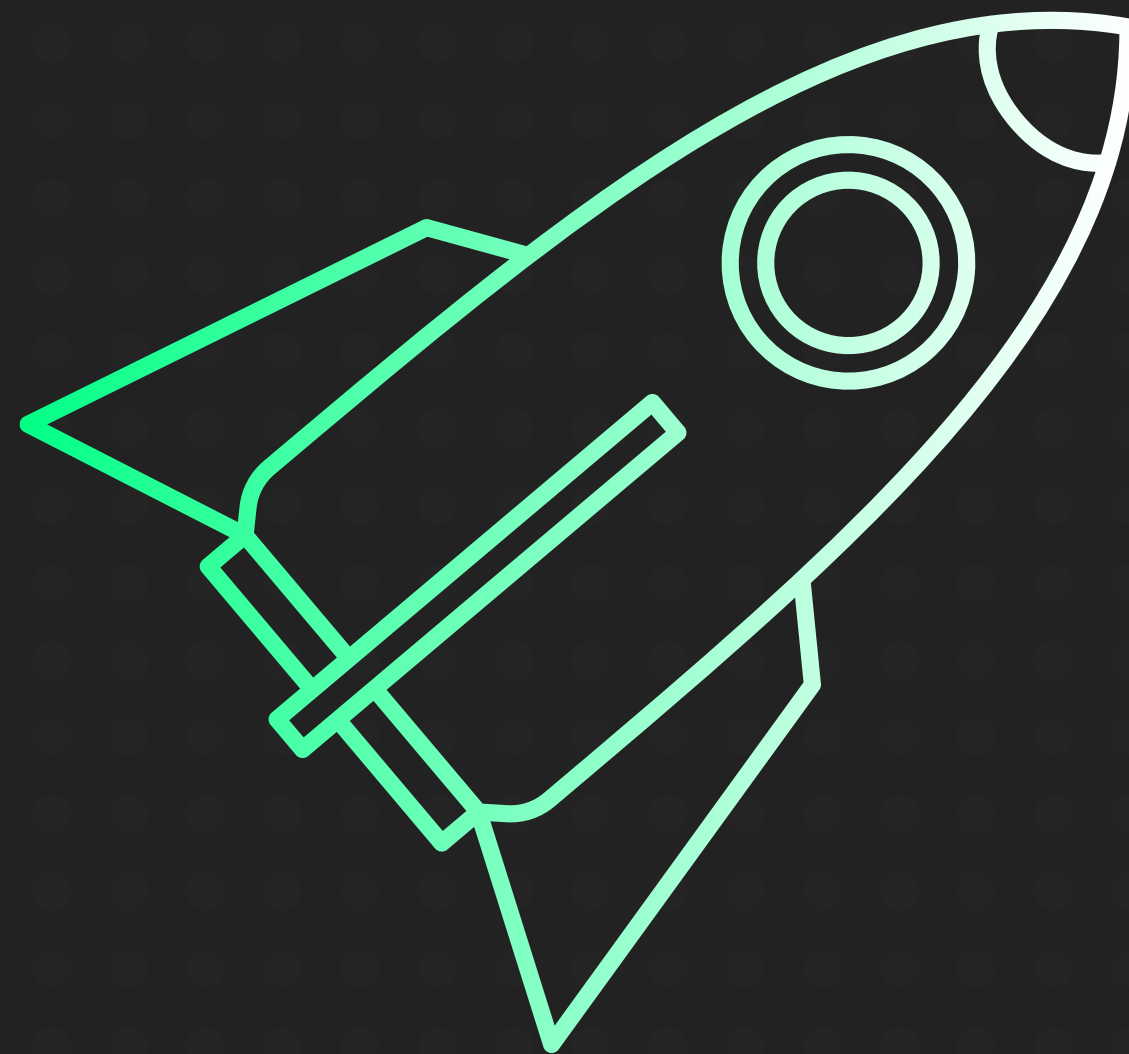
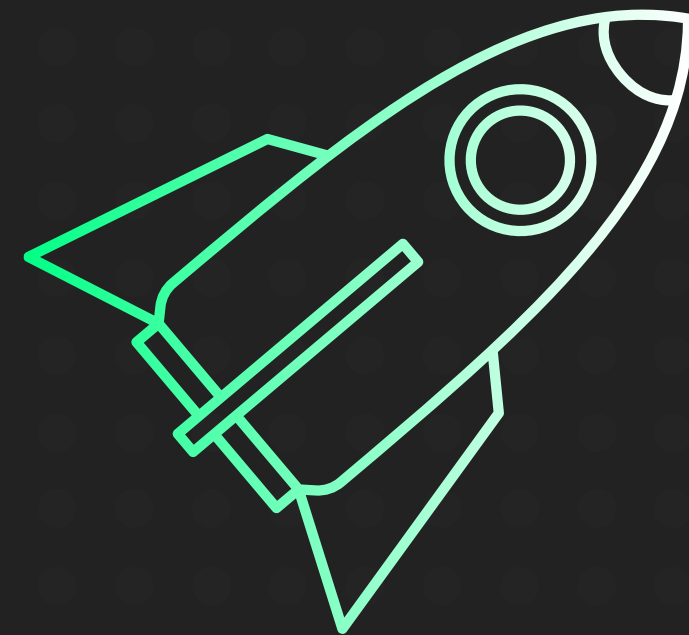


# PROJETO FINAL - COMPUTAÇÃO PARALELA



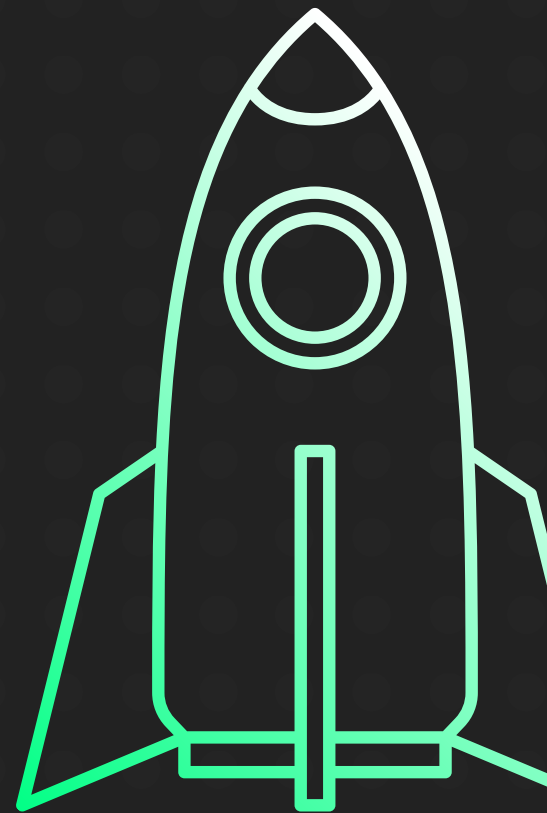
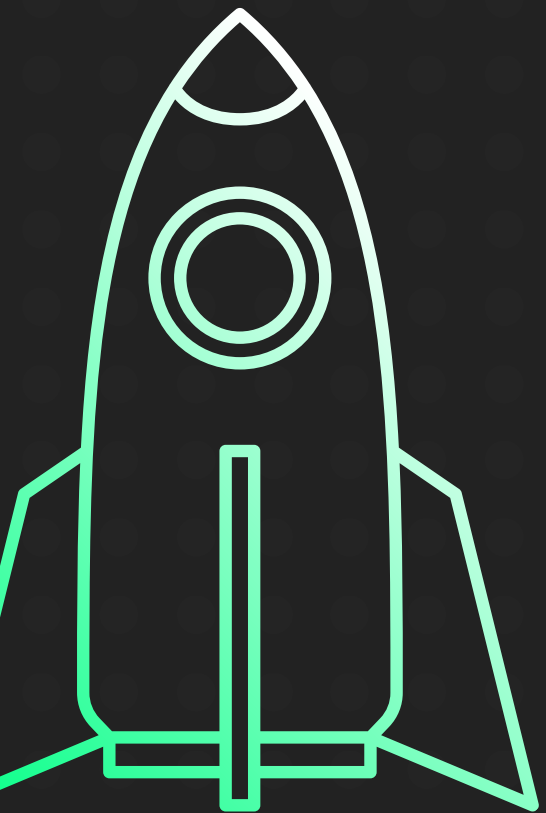
**30 de maio**

LUCAS ROCHA  
LUCAS DAMASCENO

**PROF.** EDUARDO FERREIRA DOS SANTOS

# INTRODUÇÃO

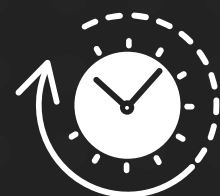
O objetivo dessa apresentação é contar um pouco do processo de desenvolvimento do nosso projeto de computação paralela durante o 5 semestre de Ciência da Computação na Universidade Presbiteriana Mackenzie. Vamos compartilhar com vocês a estratégia utilizada para calcular o número de Euler com 10.000 casas decimais.



.00  
→.0

10.000 casas decimais

1



Melhor tempo possível

2

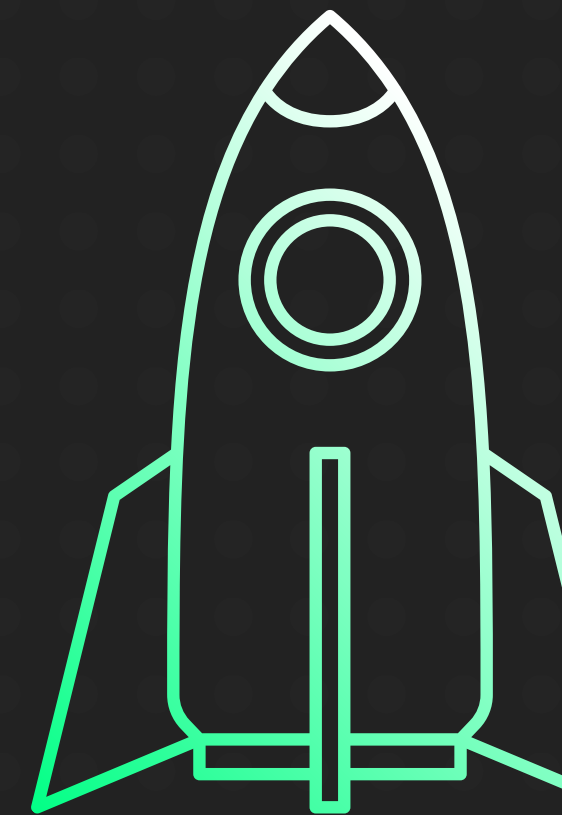
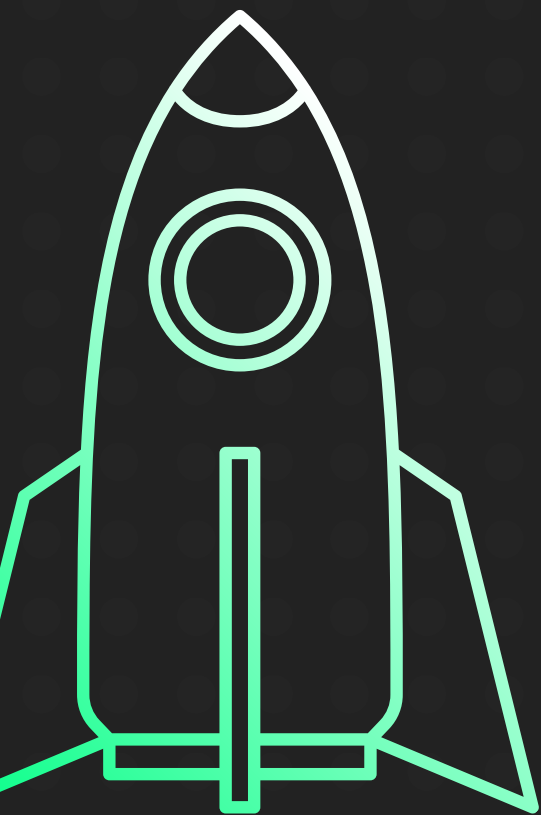
## Objetivos

Cálculo do número de Euler com 10.000 casas decimais através da série de Taylor, com foco na redução do tempo de execução por meio de paralelismo.

## **PERGUNTA 1: COMO RESOLVERAM O PROBLEMA: DESCRIÇÃO SIMPLES DO ALGORITMO E ESTRATÉGIA DE PARALELISMO ADOTADA?**

Para resolver o problema de calcular o número de Euler com 10.000 casas decimais no menor tempo possível, utilizamos um algoritmo baseado na série de Taylor e implementamos paralelismo com GMP (GNU Multiple Precision Arithmetic Library) e OMP (OpenMP).

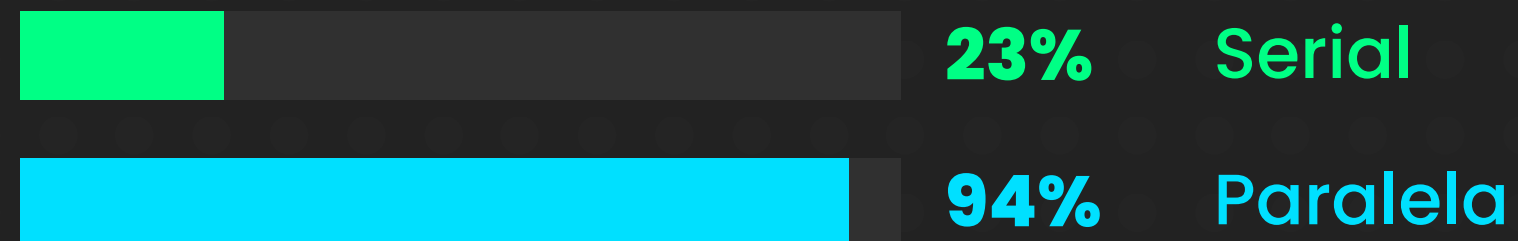
O algoritmo consiste em dividir o trabalho entre as threads disponíveis, de forma que cada thread calcula uma parte do somatório necessário para obter o número de Euler. Utilizamos uma estrutura de loop onde cada thread calcula os termos da série de Taylor de acordo com o número de iterações alocado para ela, levando em consideração o número total de threads disponíveis.




## PERGUNTA 2: QUAL FOI O SPEEDUP DA ÚLTIMA VERSÃO E COMO ELES FIZERAM PARA MELHORAR?

$$\text{Speedup} = 0.231 / 0.057 \quad \text{Speedup} \approx 4.056$$

Portanto, o speedup obtido ao utilizar a versão paralela com GMP e OMP, utilizando 2 núcleos, foi aproximadamente 4.053. Isso significa que a versão paralela foi aproximadamente 4 vezes mais rápida do que a versão serial.

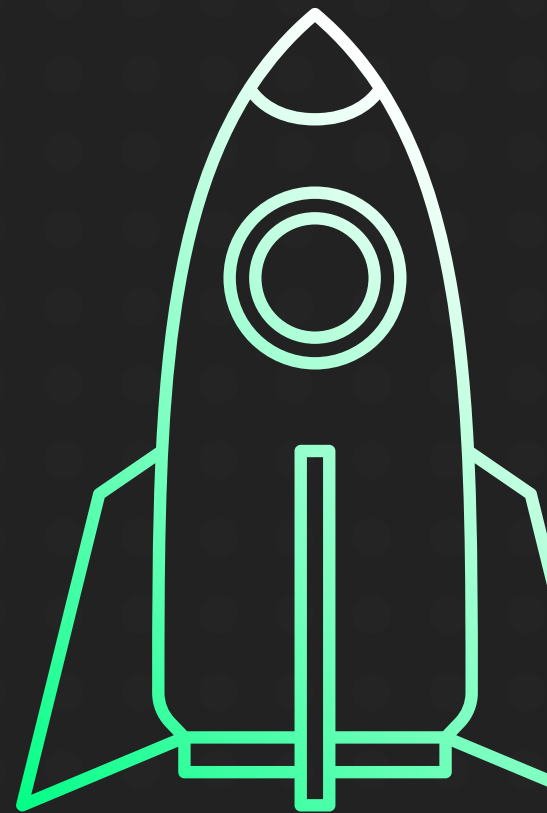
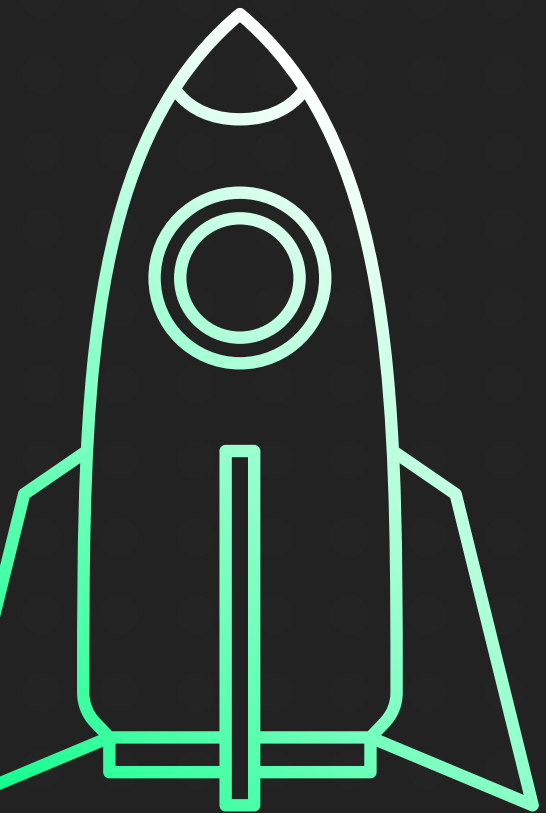





### **PERGUNTA 3: CONSIDERAÇÕES SOBRE A ESTRATÉGIA DE PARALELISMO ADOTADA E COMO ISSO AUXILIOU NA ESCALA DO PROBLEMA.**

A estratégia de paralelismo adotada com GMP e OMP foi fundamental para otimizar o tempo de execução do programa e lidar com um problema de grande escala, envolvendo o cálculo de um número com 10.000 casas decimais.

Ao dividir o trabalho entre as threads e utilizar o paralelismo, conseguimos distribuir a carga computacional de forma eficiente, permitindo que várias operações matemáticas fossem realizadas simultaneamente. Isso resultou em um aproveitamento máximo dos recursos computacionais disponíveis, acelerando significativamente o tempo de execução do programa.





# SHOW ME THE CODE!

Versão final do programa paralelo que resolve o numero de euler com GMP e OMP (2 threads):

```
int main(int argc , char* argv[]) {
    if (argc < 2){
        printf("Informe o numero de threads.");
        return 1;
    }
    int thread_count = strtol(argv[1], NULL, 10);
    int n = 33200; //130000
    mpf_t e;
    mpf_init2(e, 33500U);
    mpf_set_ui(e, 1);

    #pragma omp parallel num_threads(thread_count)
    euler(n, &e);
    gmp_printf("%.9999Ff\n", e);
    mpf_clear(e);

    return 0;
}
```

```
void euler(int n, mpf_t *e){
    int i, local_n;
    int my_rank = omp_get_thread_num();
    int thread_count = omp_get_num_threads();

    mpf_t result, e_local, fact;
    mpf_init2(result, 33500U);
    mpf_init2(e_local, 33500U);
    mpf_init2(fact, 33500U);

    mpf_set_ui(fact, 1);
    local_n = n/thread_count;
```

```
    for(i = 1; i <= local_n; i++){
        if(i > local_n*my_rank){
            mpf_mul_ui(fact, fact, i);
            mpf_ui_div(result, 1, fact);
            mpf_add(e_local, e_local, result);
        } else {
            mpf_mul_ui(fact, fact, i);
        }
    }

    #pragma omp critical
    mpf_add(*e, *e, e_local);
    mpf_clear(e_local);
    mpf_clear(result);
    mpf_clear(fact);
}
```

**RESULTADOS**

**SERIAL SEM GMP**

**0m0.001s**

**Euler:**

**2.718281828459045235**



**RESULTADOS**

**SERIAL COM GMP**

**0m0.231s**

**Euler:  
2.718281828459045...**

# RESULTADOS

PARALELO COM GMP E OMP  
(2 THREADS)

**0m0.057s**

**Euler:**  
**2.718281828459045...**

## MELHORIAS

### Sugestão de Melhoria

No código atual, o número máximo de iterações ( $n$ ) é definido como 3250. Aumentar esse valor permitiria calcular uma aproximação mais precisa do número de Euler para ainda mais casas decimais.

**MUITO  
OBRIGADO!**