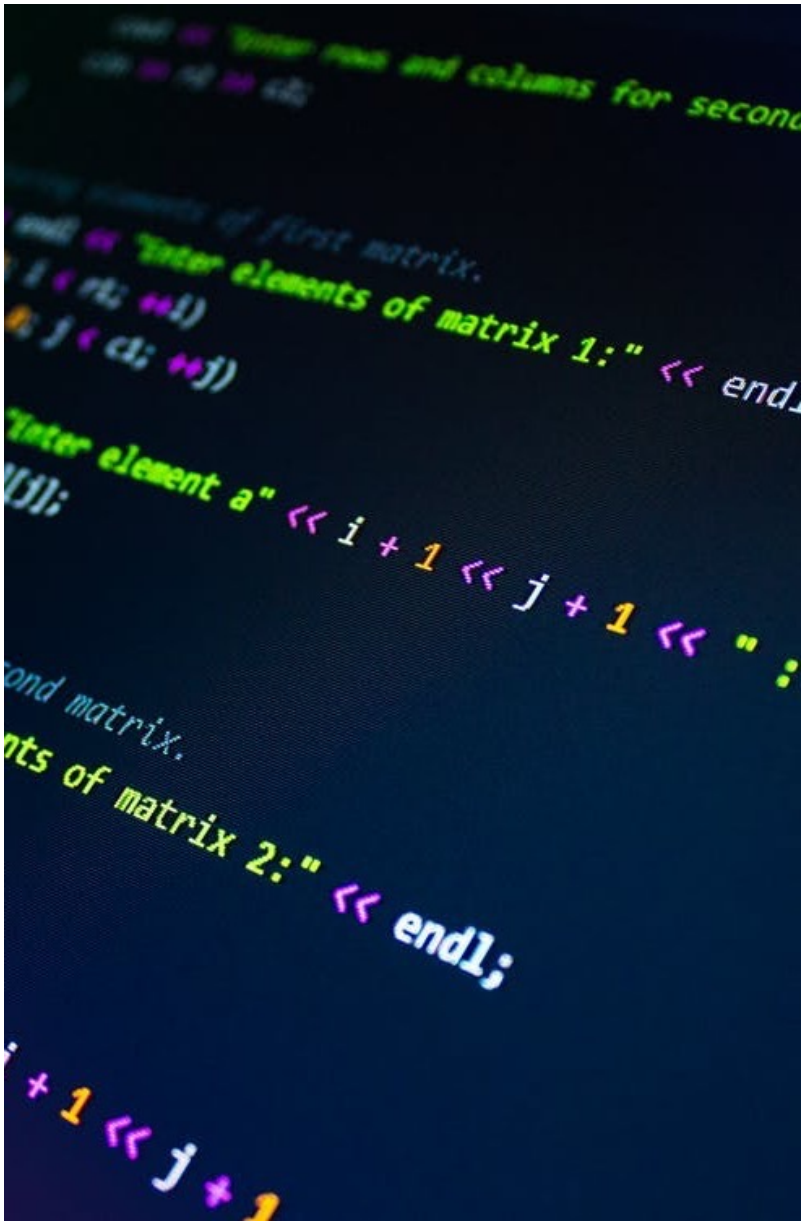
Insper

Supercomputação



Aula 04

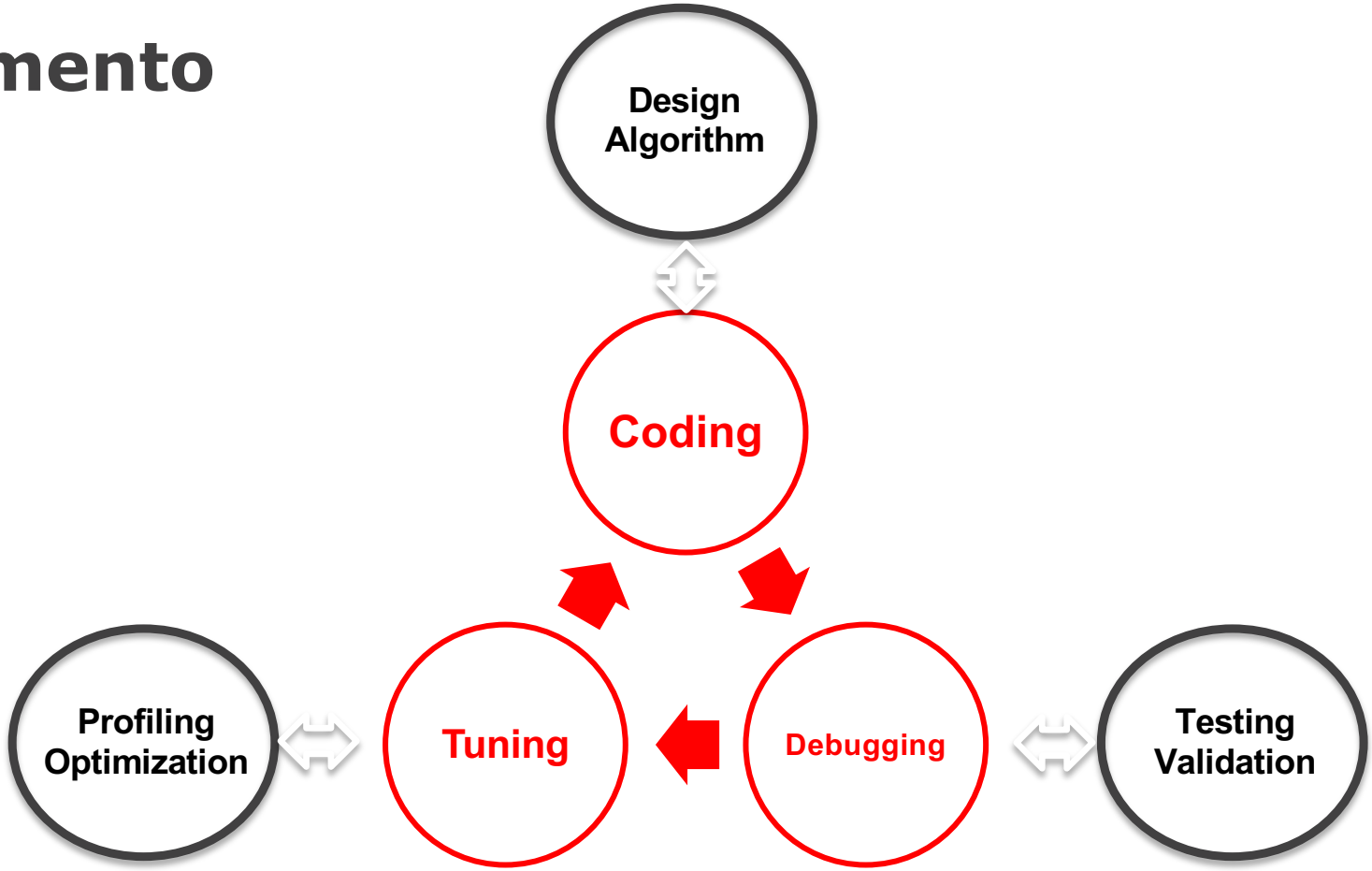
- Profiling

Vamos recordar...

- Soluções de alto desempenho se dão normalmente sob três estratégias:

1. Algoritmos eficientes
2. Implementação Eficiente
    1. Cache, paralelismo de instrução
    2. Linguagem de Programação adequada
3. Paralelismo

Ciclo habitual de desenvolvimento



Medição de desempenho

- Otimizamos um algoritmo:
  - Como medir quanto tempo cada função demora?
  - Nossa função ficou mais rápida? Se sim, quanto? Se não, por quê?
  - Como medir “quantidade de trabalho feito”?

Profiling

- Análise de um programa durante sua execução, de modo a determinar seu consumo de memória e/ou tempo de execução

- Com profiling, podemos responder duas importantes perguntas:

  - Onde o programa consome mais recursos?
  - Onde devo concentrar meus esforços de otimização?

Warm-up (roteiro)

- O problema de soma de uma matriz

- No roteiro da aula, é apresentado um código-fonte em C++ de suas funções: naive_sum e improved_sum

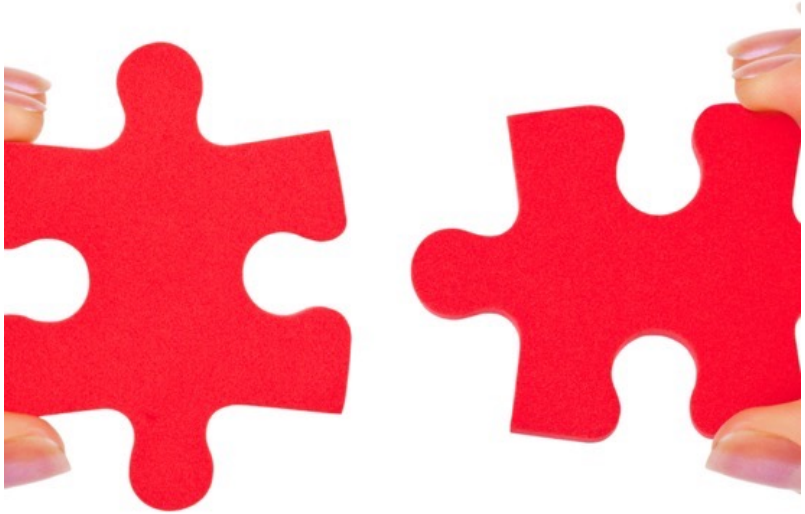
Vamos fazer uso da ferramenta **Valgrind** para realizar o profiling desse código e entender a diferença entre as duas funções

```cpp
#include<iostream>
#include<algorithm>
using namespace std;

constexpr int M = 2048;
constexpr int N = 2048;

double naive_sum(const double a[][N]){
    double sum = 0.0;
    for(int j = 0; j < N; ++j) {
        for(int i = 0; i < M; ++i)
            sum += a[i][j];
    }
    return sum;
}

double improved_sum(const double a[][N]) {
    double sum = 0.0;
    for(int i = 0; i < M; ++i)
        for(int j = 0; j < N; ++j)
            sum +=a[i][j];
    return sum;
}

int main() {
    static double a[M][N];
    fill_n(&a[0][0], M*N, 1.0 / (M*N));
    cout << naive_sum(a) << endl;
    static double b[M][N];
    fill_n(&b[0][0], M*N, 1.0 / (M*N));
    cout << improved_sum(b) << endl;
    return 0;
}
```

Roteiro

- Vamos desenvolver o roteiro da aula e conhecer as ferramentas disponíveis no Valgrind para proling de dados

Conclusões



- Entrada e saída custam caro
- Implementações diferentes do mesmo algoritmo podem ter desempenho diferentes
- Detalhes finos só são visíveis com o auxílio de ferramentas de profiling

Obrigado