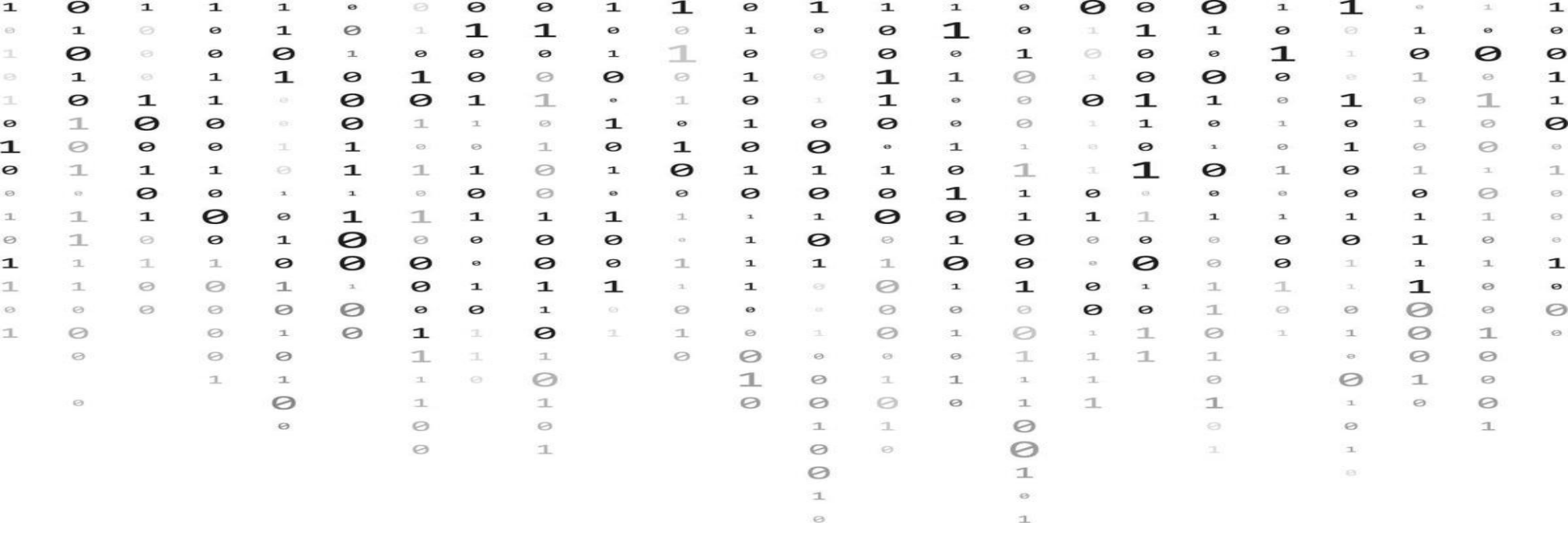
TOOLS HPC

RAINHAS DO CÓDIGO: MULHERES NA SUPERCOMPUTAÇÃO

CONHEÇA O PRINCÍPIO BÁSICO DAS FERRAMENTAS

MAÍRIAN MENDES



01

High-Performance Computing (HPC)

High-Performance Computing, refere-se ao uso de supercomputadores e técnicas de computação paralela para resolver problemas computacionalmente intensivos. Variam desde simulações científicas até a análise de big data e desenvolvimento de inteligência artificial.


MikroTik RouterOS 5.11 (c) 1999-2011 http://www.mikrotik.com/
ROUTER HAS NO SOFTWARE KEY
You have 23h30m to configure the router to be remotely accessible,
and to enter the key by pasting it in a Telnet window or in Winbox.
See www.mikrotik.com/key for more details.
Current installation "software ID": 8CJ8-ZHEX
Please press "Enter" to continue!
[admin@MikroTik] >

02

MPI (Message Passing Interface)

O MPI é uma biblioteca padrão para comunicação entre processos em um ambiente de computação paralela. Ele permite que diferentes partes de um programa rodem simultaneamente em vários processadores.

Exemplo de Código: Hello World utilizando o MPI

WOMAN_HPC - MAÍRIAN MENDES.HPC

```
# Código MPI em C

#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    printf("Hello world from processor %d out of %d processors\n", world_rank, world_size);


    MPI_Finalize();
    return 0;
}
```

Como Compilar e Executar:

WOMAN_HPC - MAÍRIAN MENDES.HPC

```
mpicc -o hello_mpi hello_mpi.c
mpirun -np 4 ./hello_mpi
```

03

Open MP (Open Multi-Processing)

Open Multi-Processing) é uma API (Application Programming Interface) que suporta programação paralela em sistemas com múltiplos processadores, permitindo a criação de programas multithread.

Exemplo de Código: Suma Paralela com OpenMP

WOMAN_HPC - MAÍRIAN MENDES.HPC

```
# Código OpenMP em C

#include <omp.h>
#include <stdio.h>

int main() {
    int n = 1000;
    int a[n], b[n], sum[n];

    // Inicializa os arrays
    for(int i = 0; i < n; i++) {
        a[i] = i;
        b[i] = i;
    }

    #pragma omp parallel for
    for(int i = 0; i < n; i++) {
        sum[i] = a[i] + b[i];
    }

    // Exibe a soma de alguns elementos
    printf("Soma dos primeiros 10 elementos:\n");
    for(int i = 0; i < 10; i++) {
        printf("%d ", sum[i]);
    }
    printf("\n");

    return 0;
}
```

Como Compilar e Executar:

WOMAN_HPC - MAÍRIAN MENDES.HPC

```
gcc -fopenmp -o sum_openmp sum_openmp.c
./sum_openmp
```

```
    // Exibe a soma de alguns elementos
    printf("Soma dos primeiros 10 elementos:\n");
    for(int i = 0; i < 10; i++) {
        printf("%d ", sum[i]);
    }
    printf("\n");

    return 0;
}
```

04

CUDA (Compute Unified Device Architecture)

É uma plataforma de computação paralela desenvolvida pela NVIDIA. Permite que os desenvolvedores utilizem a potência total das GPU’s (Unidades de Processamento Gráfico) para executar cálculos em paralelo.

Exemplo de Código: Multiplicação de Matrizes com CUDA

WOMAN_HPC - MAÍRIAN MENDES.HPC

```
# Código CUDA em C

#include <cuda.h>
#include <stdio.h>

__global__ void matrixMul(int* a, int* b, int* c, int N) {
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;

    int sum = 0;
    for (int i = 0; i < N; ++i) {
        sum += a[row * N + i] * b[i * N + col];
    }
    c[row * N + col] = sum;
}

int main() {
    int N = 2;
    int a[4] = {1, 2, 3, 4};
    int b[4] = {1, 2, 3, 4};
    int c[4] = {0};

    int *d_a, *d_b, *d_c;
    cudaMalloc((void**)&d_a, N * N * sizeof(int));
    cudaMalloc((void**)&d_b, N * N * sizeof(int));
    cudaMalloc((void**)&d_c, N * N * sizeof(int));

    cudaMemcpy(d_a, a, N * N * sizeof(int), cudaMemcpyHostToDevice);
    cudaMemcpy(d_b, b, N * N * sizeof(int), cudaMemcpyHostToDevice);

    dim3 threadsPerBlock(2, 2);
    dim3 blocksPerGrid(1, 1);
    matrixMul<<<blocksPerGrid, threadsPerBlock>>>(d_a, d_b, d_c, N);

    cudaMemcpy(c, d_c, N * N * sizeof(int), cudaMemcpyDeviceToHost);

    printf("Resultado da multiplicação de matrizes:\n");
    for (int i = 0; i < N * N; i++) {
        printf("%d ", c[i]);
        if ((i + 1) % N == 0) printf("\n");
    }

    cudaFree(d_a);
    cudaFree(d_b);
    cudaFree(d_c);

    return 0;
}
```

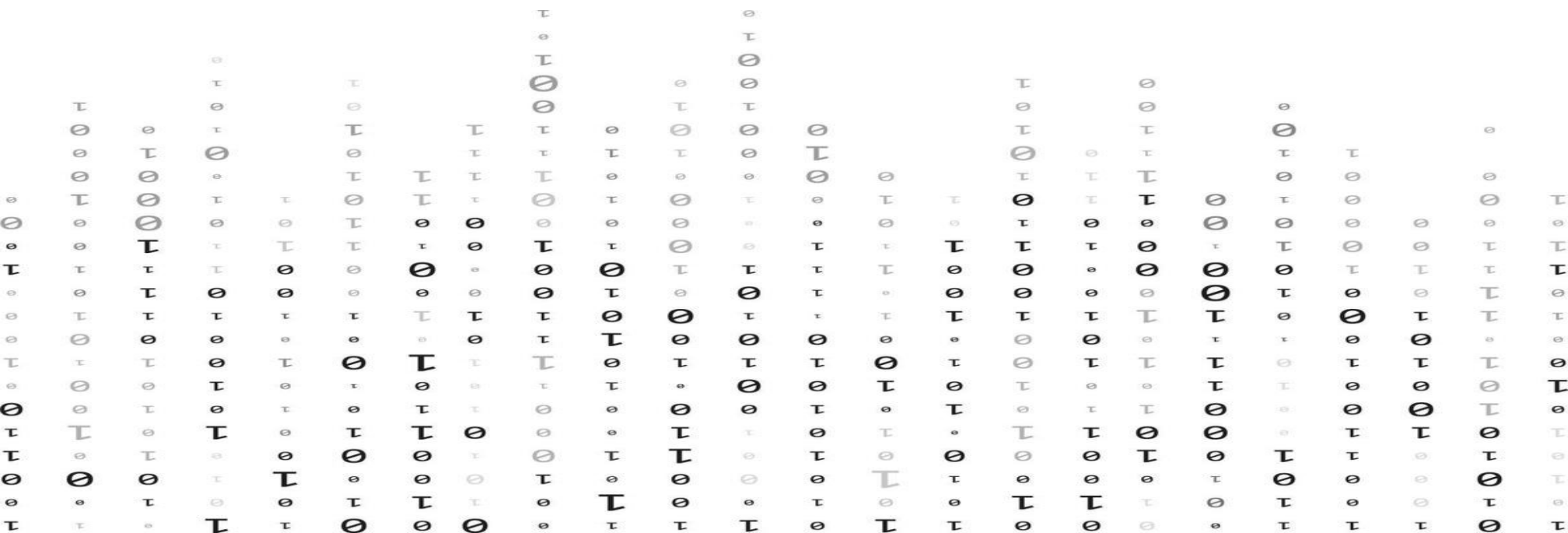
Como Compilar e Executar:

WOMAN_HPC - MAÍRIAN MENDES.HPC

```
nvcc -o matrixMul matrixMul.cu
./matrixMul
```

Conclusão:

Estas ferramentas são essenciais para o desenvolvimento de aplicações em HPC. MPI, OpenMP e CUDA permitem a programação paralela de maneira eficiente, maximizando o aproveitamento do poder computacional disponível. Com os exemplos fornecidos, você poderá iniciar sua jornada de exploração e experimentação com essas tecnologias.

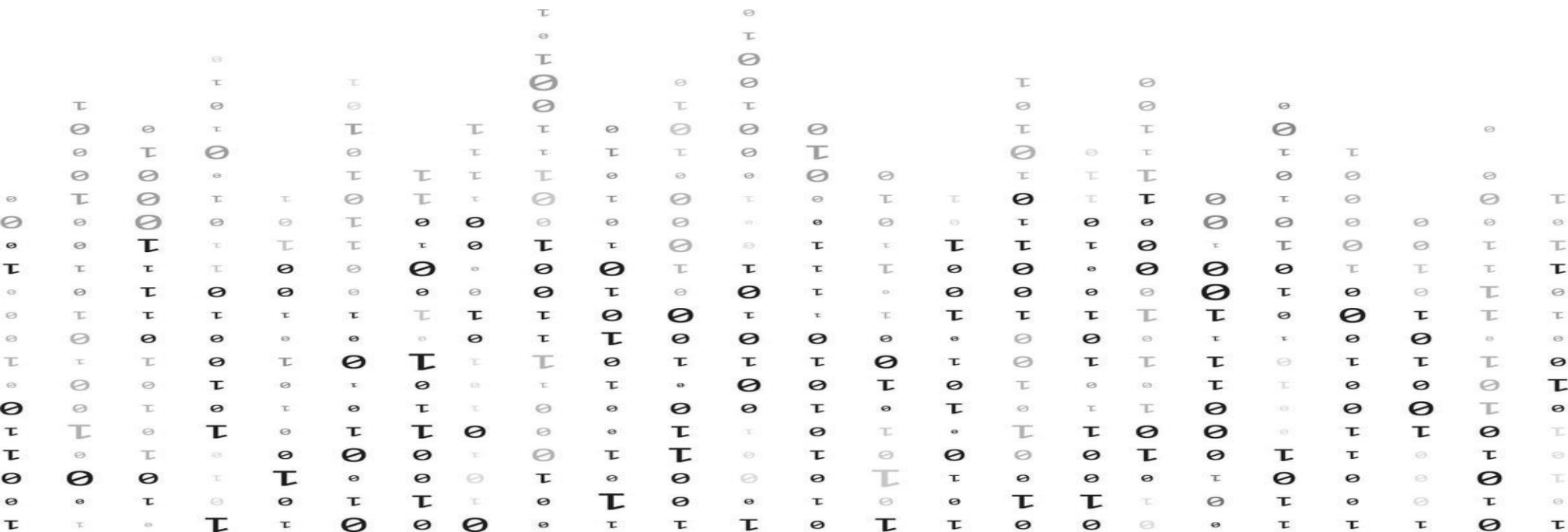


Agradecimentos

Considerações finais:

Esse e-book foi gerado por I.A. e diagramado por humano.

Este conteúdo foi gerado para fins didáticos e de construção. Não foi realizada uma validação humana cuidadosa do material aqui apresentado, portanto, é possível encontrar erros cometidos pela inteligência artificial.



Link do GitHub:

OBRIGADA POR LER ATÉ AQUI!