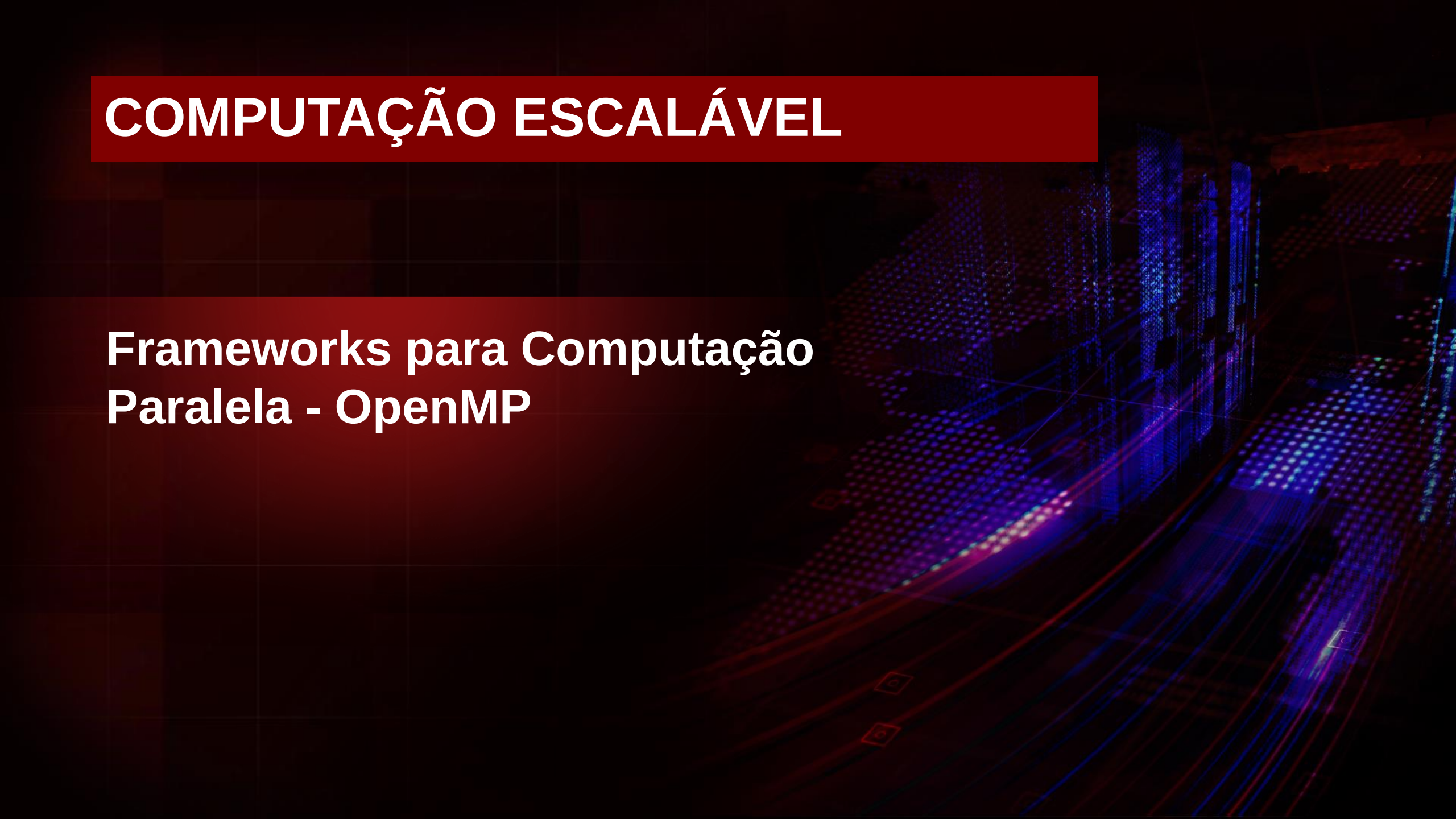
COMPUTAÇÃO ESCALÁVEL

Frameworks para Computação Paralela - OpenMP



ROTEIRO

- O que é OpenMP
- Modelo de Programação
- Modelo de Execução
- Variáveis
- Exemplo

O QUE É OPENMP?

- É um modelo de programação de memória compartilhada que nasceu da cooperação entre um grupo de grandes fabricantes de software e hardware

- API para programação paralela de arquiteturas multiprocessor foi definida inicialmente para ser usada em programas C / C++ (1998) e Fortran (1997) tanto Linux quanto Windows

- Não é uma implementação, e sim uma especificaçãoX

O QUE É OPENMP?

Objetivos

- Ser o padrão de programação para arquiteturas de memória compartilhada
- Estabelecer um conjunto simples e limitado de diretivas de programação
- Possibilitar a paralelização incremental de programas sequenciais
- Permitir implementações eficientes em problemas de granularidade fina, média e grossa

O QUE É OPENMP?

Componentes

- Bibliotecas de funções
- Diretivas de compilação
- Variáveis de ambiente

Limitações

- Quanto mais variáveis compartilhadas entre as threads, mais difícil deve ser o trabalho para garantir que a visão das variáveis são atuais – coerência de cache
- Outras limitações são baseadas em hardware e estão associadas à forma como muitos SMPs são fisicamente capazes de compartilhar o mesmo espaço de memória

O QUE É OPENMP?

- O aspecto mais difícil de criar um programa de memória compartilhada é traduzir o que se deseja para uma versão multi-thread
- Outro ponto é fazer a versão muilti-thread do programa operar corretamente

  - Sem problemas com variáveis compartilhadas
  - Situações de condições de corrida/disputa
  - Detecção e Depuração dessas condições
  - Questões relativas ao tempo de execução

MODELO DE PROGRAMAÇÃO

Paralelismo Explícito

- Cabe ao programador anotar as tarefas para execução em paralelo e definir os pontos de sincronização. A anotação é feita por utilização de diretivas de compilação embarcadas no código do programa

MODELO DE PROGRAMAÇÃO

Multithread Implícito

- Neste caso um processo é visto em conjunto de threads que se comunicam por meio da utilização de variáveis compartilhadas
  - Criação
  - Início
  - Término
- Feito de forma implícita pelo ambiente de execução sem que o programador tenha que se preocupar com tais detalhes

MODELO DE PROGRAMAÇÃO

Multithread Implícito

- Espaço de endereçamento global compartilhado entre as threads
- Variáveis podem ser compartilhadas ou privadas para cada thread
- Controle, manuseio e sincronização das variáveis envolvidas nas tarefas paralelas são transparentes ao programador

MODELO DE EXECUÇÃO

Fork-Join

- Todos os programas iniciam sua execução com um processo mestre
- O master thread executa sequencialmente até encontrar um construtor paralelo, momento em que cria um team de threads
- O código delimitado pelo construtor paralelo é executado em paralelo pelo master thread e pelo team de threads
- Quando completa a execução paralela, o team de threads sincroniza em uma barreira implícita com o master thread
- O team de threads termina a sua execução e o master thread continua a execução sequencialmente até encontrar um novo construtor paralelo
- A sincronização é utilizada para evitar a competição entre as threads

MODELO DE EXECUÇÃO

Estrutura Básica de um Programa OpenMP

```
#include <omp.h>  // incluir a biblioteca de funções OpenMP

...

main() {

  ...  // região sequencial executada apenas pelo master thread

  #pragma omp parallel  // construtor paralelo do OpenMP
  {    // master thread cria um team of threads

  ...  // região paralela executada por todos os threads

  }    // team of threads sincroniza com o master thread e termina

  ...  // região sequencial executada apenas pelo master thread

}
```

MODELO DE EXECUÇÃO

- Definições da biblioteca em omp.h
- Implementação em libgomp.so
- Compilação de um programa OpemMP
  - É preciso utilizar a opção
    - -fopenmp

VARIÁVEIS

- Declarar acesso entre as threads
  - Private
    - Pertencem e são conhecidas apenas a cada segmento específico

```
int id, nthreads;
pragma omp parallel private(id)
{id = omp_get_thread_num();
```

VARIÁVEIS

- Declarar acesso entre as threads
  - Shared
    - Variáveis compartilhadas são conhecidas por todas as threads
    - Cuidados devem ser tomados quando utilizadas essas variáveis, já que a manipulação incorreta poderá dificultar a detecção de erros como condições de disputa e deadlocks

```
int id, nthreads, A, B; A = getA();
        B = getB();
        #pragma omp parallel private(id,nthreads) shared(A,B)
        { id = omp_get_thread_num();
```

VARIÁVEIS

- Inicialização de Variáveis
  - Firstprivate
    - Permite inicializar uma variável privada na thread master antes de entrar em uma região paralela do código. Caso contrátrio, todas as variáveis privadas serão consideradas não inicializadas no início da thread

```
int id, myPi; myPi = 3.1459
        #pragma omp parallel private(id) firstprivate(myPi)
        { id = omp_get_thread_num();
```

MAPEAMENTO PARA THREADS

```
int a, b;
main() {
    // serial segment
    #pragma omp parallel num_threads (8) private (a) shared (b)
    {
        // parallel segment
    }
    // rest of serial segment
}
```

Sample OpenMP program

Code inserted by the OpenMP compiler

```
int a, b;
main() {
    // serial segment

    for (i = 0; i < 8; i++)
        pthread_create (......, internal_thread_fn_name, ...);

    for (i = 0; i < 8; i++)
        pthread_join (.....);

    // rest of serial segment

}

void *internal_thread_fn_name (void *packaged_argument) [
    int a;

    // parallel segment

}
```

Corresponding Pthreads translation

EXEMPLO

Hello World!

```
#include <omp.h>
main () {
   int nthreads, tid; // Fork team-threads, cada uma cópia das variáveis.

#pragma omp parallel private(nthreads, tid)
{                                          // Obtem e escreve thread-id
   tid = omp_get_thread_num();
   printf("Hello World from thread = %d\n", tid);
                                           // apenas o master thread faz isto
   if (tid == 0){
        nthreads = omp_get_num_threads();
        printf("Number of threads = %d\n", nthreads);
   }
} /* Todos os theards juntam-se no master
}
```

BIBLIOGRAFIA

1. http://www.openmp.org
2. Introduction to Parallel Computing, Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar - 2ª ed., Addison Wesley
3. OpenMP - https://hpc-tutorials.llnl.gov/openmp/