Introdução ao OpenMP

Open Multi**P**rocessing

O que é OpenMP?

- OpenMP é uma **interface de programação** (API), portável, baseada no **modelo de programação paralela de memória compartilhada** para arquiteturas de múltiplos processadores.

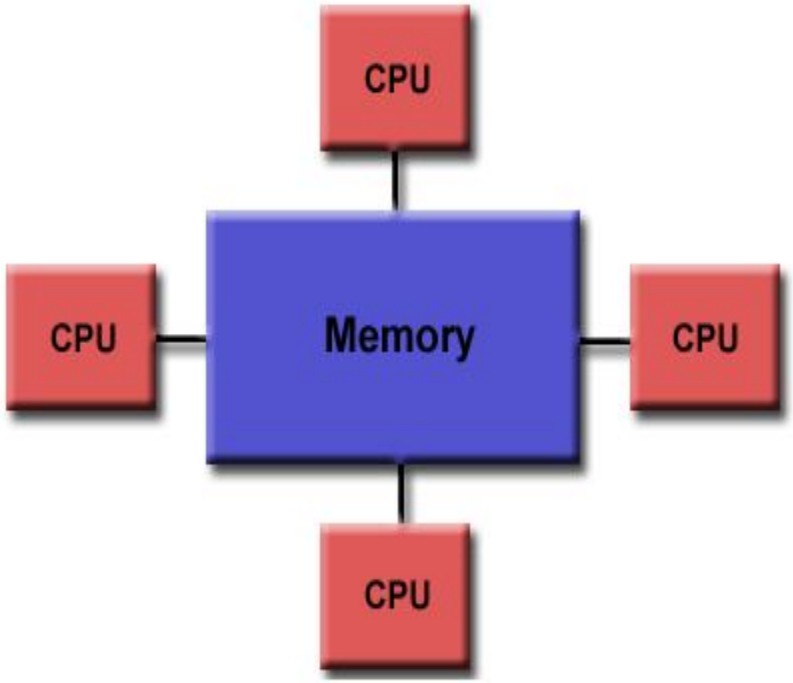
  – É composto por três componentes básicos:
  Diretivas de Compilação;
  Biblioteca de Execução;
  Variáveis de Ambiente.

OpenMP

- Definido e mantido por **um grupo composto na maioria por empresas de hardware e software**, denominado como OpenMP ARB (Architecture Review Board).

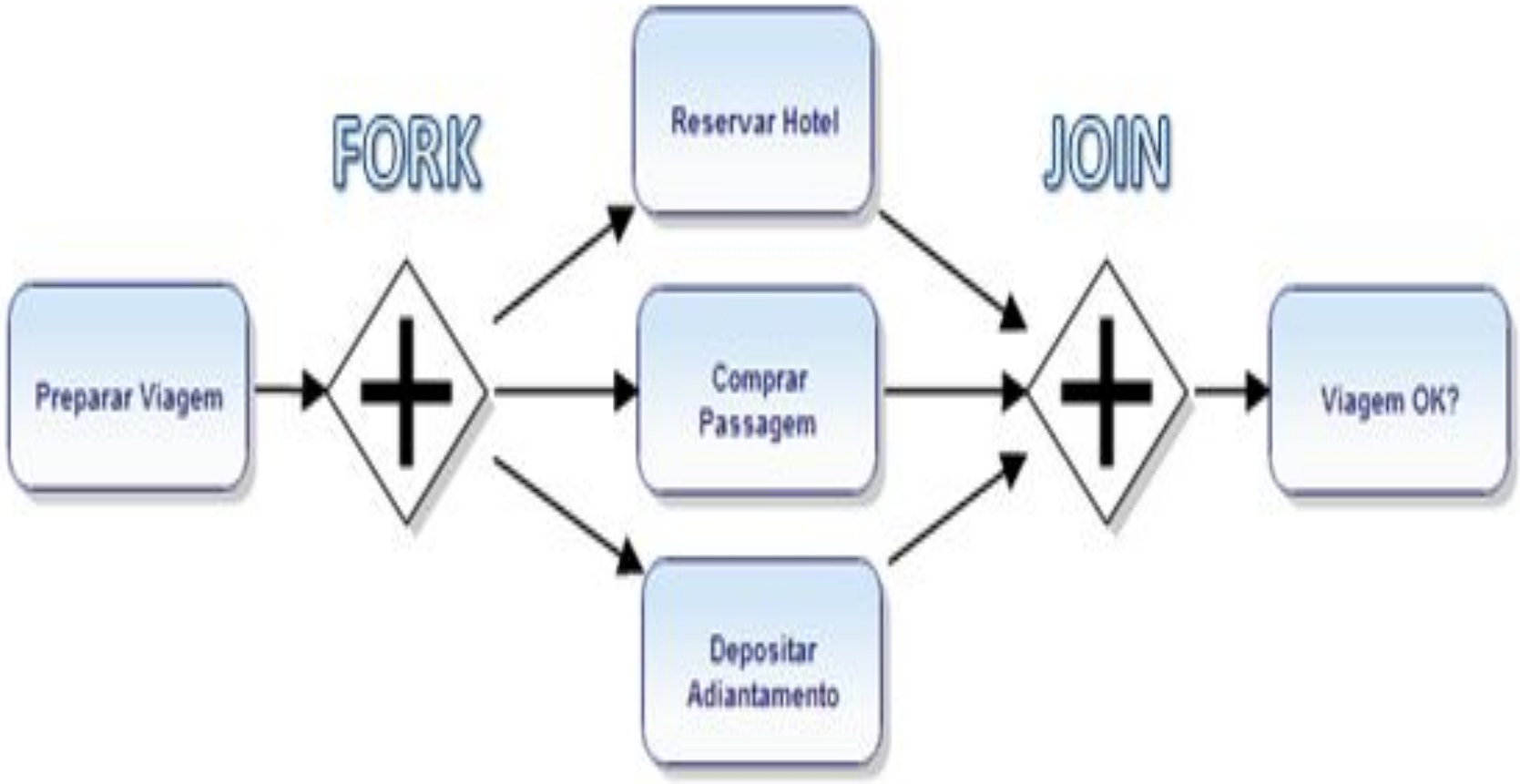
Memória Compartilhada

Ambiente com vários processadores que compartilham o espaço de endereçamento de uma única memória.

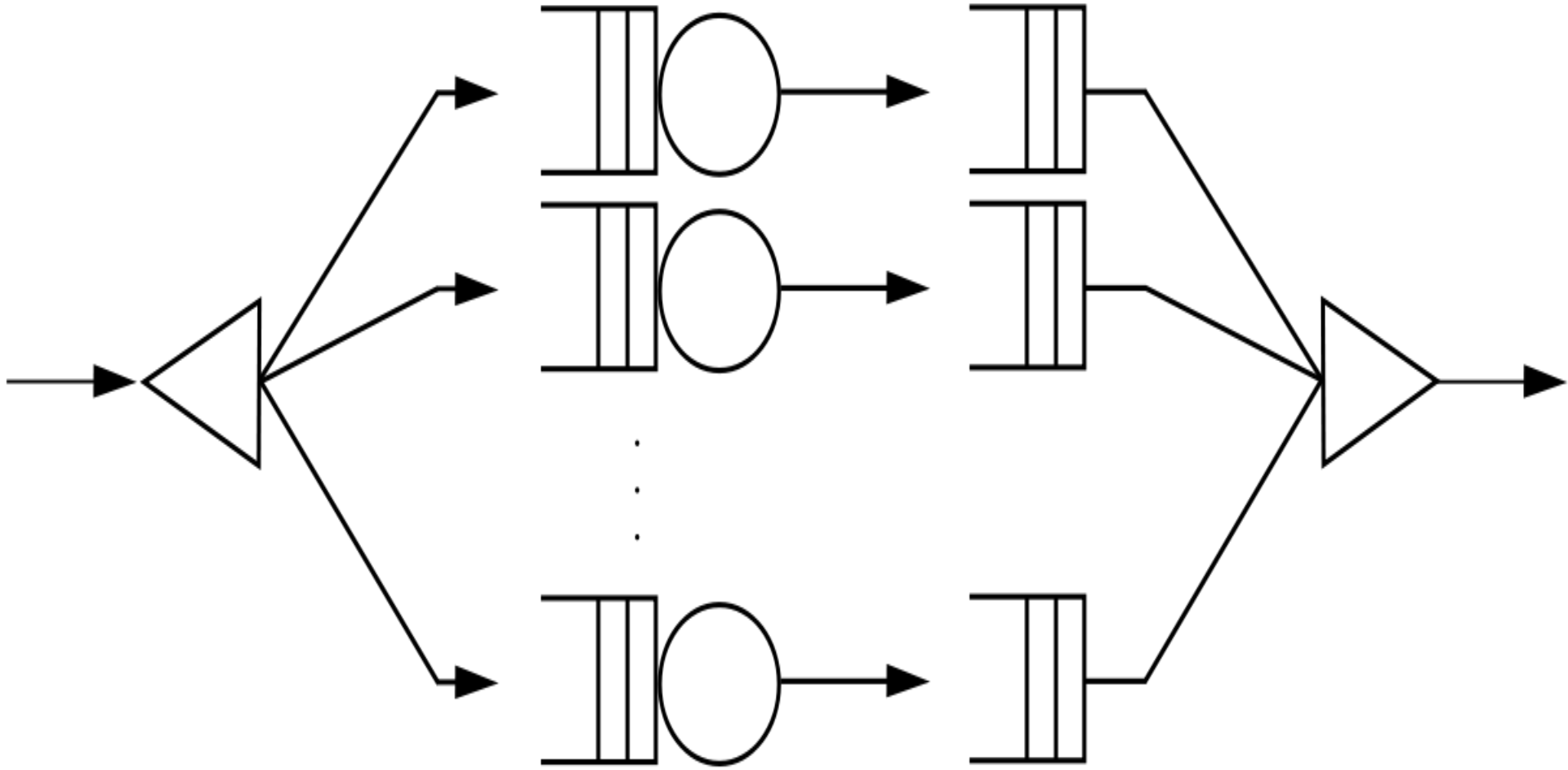


Os processos compartilham um espaço de endereçamento comum, no qual o acesso é feito no modo assíncrono; não há necessidade de especificar explicitamente a comunicação entre os processos. A implementação desse modelo pode ser feita pelos compiladores nativos do ambiente.

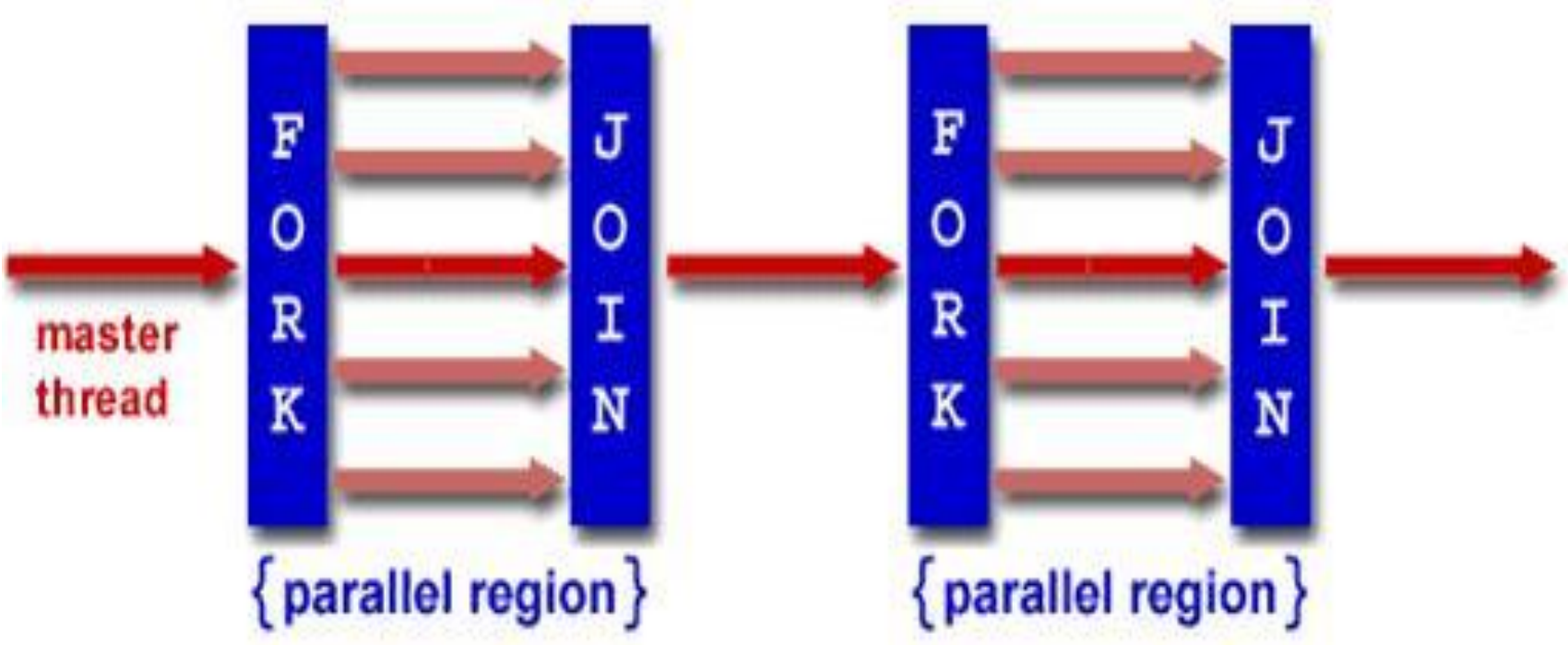
Exemplo de Tarefas Paralelas



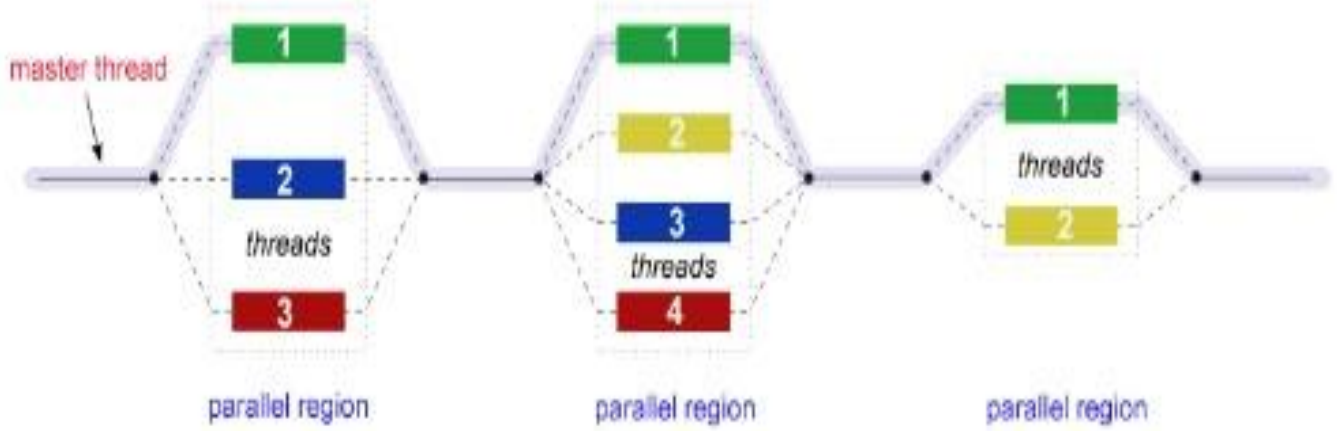
Modelo de Execução Fork-Join



Fork-Join e Regiões Paralelas



Modelo de Execução Fork - Join



OpenMP usa o modelo fork-join para paralelizar a execução

FORK: A thread principal cria um grupo de ***threads*** paralelas;

JOIN: Quando grupo threads completa a região paralela, ele sincroniza e finaliza, mantendo apenas a thread principal.

Modelo de Execução Fork-Join

1. Todos os programas OpenMP começam como um processo simples: “Thread Mestre”. A “thread mestre” executa sequencialmente até a primeira definição de uma **região paralela** ser encontrada.

   Quando uma “thread mestre” chega a uma definição de região paralela, ela cria um conjunto de “threads”.

   A “thread mestre” faz parte do conjunto de “threads” e possui o número de identificação “0”.

Modelo de Execução Fork-Join

2. “**FORK**” (bifurcar): O thread mestre cria um conjunto (”team”) de threads paralelos.

3. Os comandos de um programa que estão inseridos na região paralela serão executados em paralelo pelas diversas “threads” criadas.

Modelo de Execução Fork-Join

4. “**JOIN**” (unir): Quando o conjunto de “threads” completar a execução dos comandos na região paralela, as “threads” são sincronizadas e finalizadas, permanecendo apenas a “thread” mestre em execução.

   Existe um ponto de sincronização (“**barreira**”) no final de uma região paralela, sincronizando o fim de execução de cada “thread”. Somente a “thread” mestre continua desse ponto.

OpenMP

- Baseado em diretivas de compilação.

- Todo **paralelismo do OpenMP** é especificado através de **diretivas de compilação**.

Exemplo de estrutura OpenMP - C / C++

```
#include <omp.h>
main () {
int var1, var2, var3;
*** Código serial
. . .

*** Início da seção paralela. “Fork” um grupo de “threads”.

#pragma omp   parallel    private(var1, var2)   shared(var3)
    {
      *** Seção paralela executada por todas as
          “threads”
       . . .

      *** Todas as “threads” efetuam um “join” à thread mestre e finalizam
    }
*** Código serial
     . . .
     . . .
}
```

Formato C/C++

#pragma omp Diretiva [atributos, …]

Um Exemplo

#pragma omp parallel default(shared) private(beta,pi)

Segue as regras de sintaxe do C/C++

“Case sensitive” ("sensível a maiúsculas e minúsculas"**)**

Linhas de diretivas muito longas podem continuar em diversas linhas, acrescentando o caractere de nova linha (“\”) no final da diretiva.

Construção Parallel em C/C++

Identifica um bloco de código que será executa por múltiplos “threads”. É a construção fundamental do OpenMP.

#pragma omp parallel [atributo ...] nova_linha
if (expressão lógica)
private (lista)
shared (lista)
default (shared | none)
firstprivate (lista)
reduction (operador: lista)
copyin (lista)

estrutura de bloco

Atributos:

IF - Especifica uma condição para que o grupo de "threads" seja criado, se o resultado for verdadeiro.

Se falso, a região definida como paralela, será executada somente pela "thread" mestre.

-O número de “threads”

Em uma execução com o OpenMP, o número de “threads” é determinado pelos seguintes fatores, em ordem de precedência:

1. Utilização da função omp_set_num_threads() no código C/C++;

2. Definindo a variável de ambiente OMP_NUM_THREADS, antes da execução;

3. Implementação padrão do ambiente: número de núcleos processadores em um nodo.

Regiões paralelas recursivas

Uma região paralela dentro de outra região paralela resulta na criação de um novo grupo de “threads” de apenas uma “thread”, por “default”.

É possível definir um número maior de “threads”, utilizando um dos dois métodos disponíveis:

1. Utilização da função omp_set_nested()no código C/C++;

2. Definindo a variável de ambiente OMP_NESTED, antes da execução.

“Threads” dinâmico

Um programa com várias regiões paralelas, utilizara o mesmo número de “thread” para executar cada região.

Isso pode ser alterado permitindo que durante a execução do programa, o número de “threads” seja ajustado dinamicamente para uma determinada região paralela.

Dois métodos disponíveis:

1. Utilização da função omp_set_dynamic() no código C/C++;
2. Definindo a variável de ambiente OMP_DYNAMIC, antes da execução.

```
#include <omp.h>
main () {
int nthreads, tid;

/* Fork a team of threads giving them their own
copies of variables */

#pragma omp parallel private(nthreads, tid)
{
/* Obtain and print thread id */
tid = omp_get_thread_num() ;
printf("Hello World from thread = %d\n", tid);

 /* Only master thread does this */
if (tid == 0)
   {
     nthreads = omp_get_num_threads();
     printf("Number of threads = %d\n",
                nthreads);
   }
}
/* All threads join master thread and terminate */
}
```

Atributos

PRIVATE - Declara que as variáveis listadas serão de uso específico de cada “thread”. Essas variáveis não são iniciadas.

SHARED (“default”) - Declara que as variáveis listadas irão compartilhar o seu conteúdo com todas as “threads” de um grupo. As variáveis existem em apenas um endereço de memória, que pode ser lido e escrito por todas as “threads” do grupo.

DEFAULT - Permite que o programador defina o atributo “default” para as variáveis em uma região paralela (PRIVATE, SHARED ou NONE).

Atributos Comuns

Em programação OpenMP é importante entender como é utilizado o dado em uma “thread” (“Data scope”).

Devido ao fato do OpenMP ser baseado no modelo de programação de memória compartilhada, a maioria das variáveis são compartilhadas entre as “threads” por definição (Variáveis Globais).

Atributos Comuns

Variáveis globais, são as variáveis definidas

C: Variáveis estáticas

Variáveis não compartilhadas, locais, privadas:

Variáveis de índice (“loops”)

Os atributos comuns são utilizados por diversas diretivas (PARALLEL, DO/for, e SECTIONS) para controlar o acesso as dados.

Definem **como os dados das variáveis serão transferidos das seções seriais para as seções paralelas**.

Definem **quais as variáveis serão visíveis por todas as “threads” e quais serão locais**.