Introdução ao MPI

História

- Cooperação entre Universidades, Empresas e utilizadores dos EUA e Europa (1992)

- Publicação do padrão MPI (1994)

Várias implementações

- OpenMPI
- Intel MPI
- LAM/MPI
- MPICH
- CHIMP
- LA-MPI
- FT-MPI

Versões

- MPI 1
  - Comunicação ponto a ponto
- MPI 2
  - Criação de processos em tempo de execução
- MPI 3
  - Operações não-bloqueantes sobre conjuntos
- MPI 4
  - Comunicação particionada, persistência

O que é MPI?

- É um padrão para comunicação em programas paralelos
- Biblioteca de Message Passing definida para comunicação entre processos
- Fornece rotinas para que os processos se comuniquem
- O paralelismo é explícito

Como funciona?

- Problemas são divididos em partes
- As partes são distribuídas entre as máquinas, para processamento
- Os resultados são enviados à uma máquina receptora
- A máquina receptora coleta, agrupa e fornece o resultado desejado

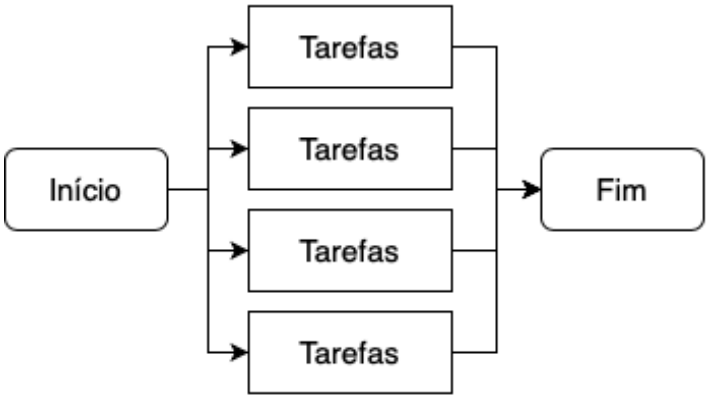
Paralelismo

Programação Sequencial

Paralelismo

Programação Sequencial

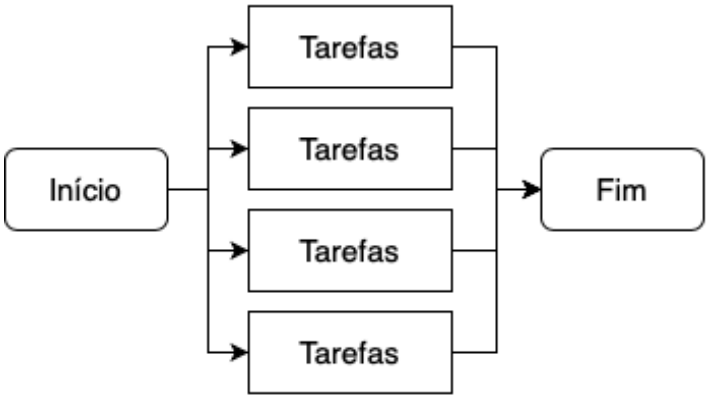
Programação Paralela



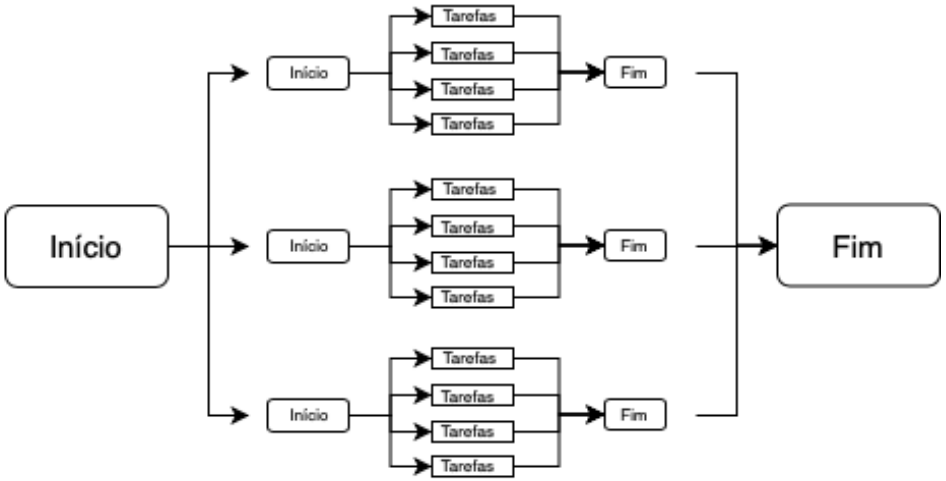
Paralelismo

Programação Sequencial

Programação Paralela



Programação Paralela e Distribuída



SPMD

- Single Program, Multiple Data
- Cada processo executa exatamente o mesmo programa
- Os dados são divididos e distribuídos aos processos participantes

SPMD

- Através de condições de teste sobre o *rank* dos processos, diferentes processos executam diferentes partes do programa

Outros modelos

- SPMD é o mais usual
- SPSP (Single Program, Single Data)
- MPMD (Multiple Program, Multiple Data)
- MPI não limita o modelo de programação

Estrutura do código MPI

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

int main(int argc, char **argv)
{
MPI_Init(&argc, &argv);                  // Inicia o MPI, aceitando os parametros como argumento
MPI_Comm_size(MPI_COMM_WORLD, &total);   // Recebe quantidade total de processos
MPI_Comm_rank(MPI_COMM_WORLD, &myid);    // Recebe o identificador desse processo

if (myid == 0)                           // Se o meu identificador eh 0, sou o primeiro processo
   {
   ...
}
else                                     // Se nao for o primeiro processo
   {
  ...
}
MPI_Finalize();                          // Funcao para finalizar o MPI
}
```

Troca de Mensagens no MPI

Conceitos básicos

- **Processo**: cada parte dividida da aplicação. Podem ser executados em uma única máquina ou em várias.

- **Rank**: identificação única dos processos atribuída pelo sistema operacional. Vai de 0 até n-1. O rank é utilizado para enviar e receber mensagens.

Conceitos básicos

- **Mensagem**: conteúdo de uma comunicação. Formada de duas partes: endereço e dados.

- **Grupo**: conjunto de processos que podem se comunicar entre si. Sempre está associado a um comunicador.

Conceitos básicos

- **Comunicador**: um objeto local que representa o domínio de uma comunicação.

- **MPI_COMM_WORLD**: comunicador predefinido que inclui todos os processos definidos pelo usuário em uma aplicação MPI.

Comunicação entre processos

- Buffering
- Assíncrona
- Síncrona
- Bloqueante
- Não-bloqueante

Buffering

- A mensagem a ser enviada é copiada para um buffer local do programa
- O processo emissor não fica dependente da sincronização com o processo receptor
- Deve-se definir explicitamente o buffer associado

Síncrona

- O envio da mensagem só ocorre quando houver a confirmação de que o receptor está pronto para receber

- Até o recebimento da confirmação, o processo emissor fica na espera

Assíncrona

- Mensagem é enviada sem nenhuma confirmação do receptor
- A mensagem pode ficar pendente no ambiente de execução durante algum tempo

Bloqueante

- A execução do programa é interrompida enquanto a comunicação não acontece

Não-bloqueante

- A execução do programa é independente do sucesso da comunicação

Principais rotinas

- Rotinas de inicialização e encerramento
- Rotina de grupos
- Rotinas de envio e recebimento de mensagens

Inicialização e encerramento

- MPI_Init(int *argc, char ***argv)

  mpirun –np 4 programa

- MPI_Finalize(void)

Rotinas de grupos

- MPI_Comm_rank
  - Identifica o processo dentro de um determinado grupo
- MPI_Comm_size
  - Retorna o número de processos dentro do grupo

Envio bloqueante

- MPI_Send(buf, count, datatype, dest, tag, comm)
  - IN buf: endereço do buffer de envio
  - IN count: número de elementos a enviar
  - IN datatype: tipo dos elementos a enviar
  - IN dest: identificador do processo de destino
  - IN tag: (etiqueta) da mensagem
  - IN comm: communicator (handle)

Recebimento bloqueante

- MPI_Recv(buf, count, datatype, source, tag, comm,status)
  - OUT buf: endereço do buffer de recebimento
  - IN count: número de elementos a receber
  - IN datatype: tipo dos elementos a receber
  - IN dest: identificador do processo de origem
  - IN tag: (etiqueta) da mensagem
  - IN comm: communicator (handle)

Sincronização

- Implícita
  - Através de comunicação bloqueante
- Explícita e global
  - Comunicação não-bloqueante
    - MPI_Isend
    - MPI_Irecv
  - Uso de barreiras
    - MPI_Barrier

OpenMPI

- Compilar
  - mpicc programa.c –o programa
- Executar
  - mpirun –np 4 programa
  - mpirun --hostfile arquivo.txt programa