① 9ª Aula (8ª Aula : 1ª Avaliação)

Sistemas distribuídos

→ Coleção de entidades independentes (nós) que cooperam entre si, através de comunicação, para resolver um problema que não pode ser resolvido individualmente por cada nó

→ Exemplos na natureza
* Cardume de peixes - "A união faz a força"
* Voo em formação dos pássaros - "Economia de energia"
* etc ...

②

→ Um sistema distribuído existe sempre que houver comunicação entre nós autónomos (agentes inteligentes) e geograficamente distribuídos

→ Visão geral:

③ Características de sistemas distribuídos

- Ausência de um relógio global
- Ausência de memória compartilhada
- Separação geográfica e incerteza em tempos de comunicação e processamento
- Possibilidade de falhas independentes de nós e da rede
- Dificuldade (impossibilidade) de conhecer o estado global do sistema

④ Algoritmos distribuídos

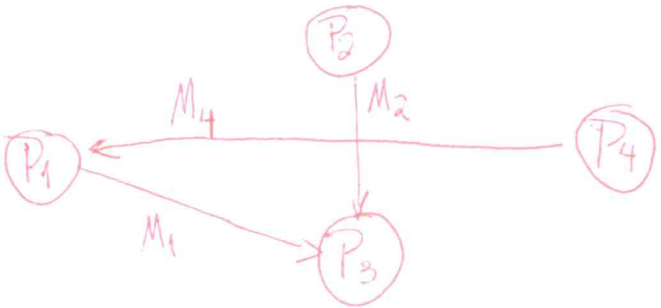
- Algoritmos distribuídos são programas que executam em nós de uma rede (com atrasos e confiabilidade não conhecidos)
- É composto por processos que conjuntamente e de forma ordenada, realizam uma tarefa e precisam manter a consistência

(5) Premissas desejáveis

→ O que espetamos de um algoritmo?
  * Estar correto
  * Ter baixa complexidade

→ Em algoritmos distribuidos tudo depende do ambiente de execução:
  * Grau de assincronismo
  * Garantia de entrega de mensagens
  * Possibilidade de falhas (nós e rede)

(6) Comunicação entre processos (interprocessos)



Pi é um processo que:
→ encapsula um estado (variáveis com valores)
→ reage a eventos externos
→ interage através do envio de mensagens

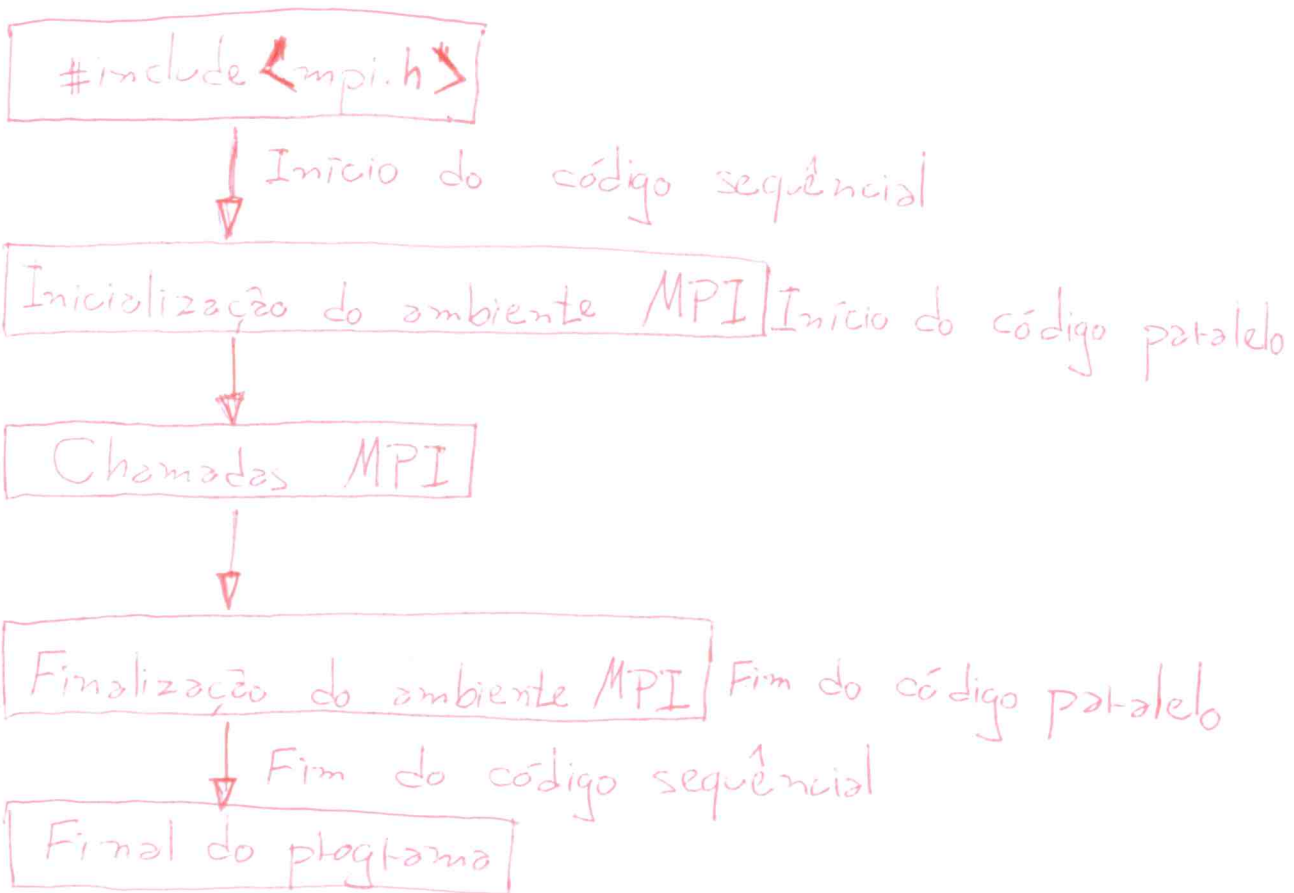
⑦ Comunicação em sistemas distribuídos

- Sistema centralizado
  - * Memória compartilhada
- Sistema distribuído
  - * Ausência de memória compartilhada
  - * Comunicação por troca de mensagens
- Modelo cliente-servidor
  - * Servidor fornece serviços aos clientes
  - * Cliente consome serviços dos servidores
  - * Simplicidade

⑧ Modelo de programação com MPI

- MPI = Message Passing Interface
- SPMD (Single Program Multiple Data)
- Um processador/processo gerente envia e gerencia um "mesmo programa" em todas as máquinas do sistema distribuído
- Implementa primitivas de comunicação ponto-a-ponto e coletivas
- Abstrai aspectos de protocolos de transporte

(9) Estrutura de um programa MPI



(10) Meu primeiro programa com MPI

Open MPI ou MPICH ?

```
#include <mpi.h>
#include <stdio.h>
int main (int argc, char* argv[])
{
 int rank, size;
 MPI_Init (&argc, &argv); // inicializa o MPI
 MPI_Comm_rank (MPI_COMM_WORLD, &rank); // id do processo corrente
 MPI_Comm_size (MPI_COMM_WORLD, &size); // número de processos
 printf("Olá do processo %d de %d\n", rank, size);
 MPI_Finalize(); // finaliza o MPI
}
```

(11) Compilação e execução

- Para compilar

  mpicc mpi_ola.c -o ola

- Para executar

  mpirun -np 5 ./ola

Observação: rank $\in \{0, 1, \ldots, N-1\}$ considerando N(size) processos

(12) Troca de mensagens com MPI

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

int size, rank, msg; source, dest, tag;
int main (int argc, char* argv[]){
    MPI_Status stat;

    MPI_Init (&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    :
    :
```

(13)

```
...
if(rank == 0){
  msg=42; dest=1; tag=0;
  MPI_Send(&msg, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
  printf("Processo %d enviou %d para o processo %d\n", rank, msg, dest);
}
if(rank == 1){
  source = 0; tag = 0;
  MPI_Recv(&msg, 1, MPI_INT, source, tag, MPI_COMM_WORLD, &stat);
  printf("Processo %d recebeu %d do processo %d\n", rank, msg, source);
}
MPI_Finalize();
}
```

(14) Primitivas básicas de comunicação

```
MPI_Send(void *buf, int count, MPI_Datatype datatype,
         int dest, int tag, MPI_Comm comm)
```

- void *buf: ponteiro para mensagem que será enviada
- int count: quantidade de elementos a serem enviados
- MPI_Datatype datatype: constante que define o tipo de dado que será enviado
- int dest: rank do nó destino
- int tag: rótulo da mensagem
- MPI_Comm comm: comunicador

(15)

MPI_Recv(void *buf, int count, MPI_Datatype datatype,
int source, int tag, MPI_Comm comm,
MPI_Status *status)

- void *buf: endereço da variável que receberá os dados
- int count: número de elementos a serem recebidos
- MPI_Datatype datatype: tipo de dado que será recebido
- int source: rank do nó remetente
- int tag: rótulo da mensagem (conforme envio)
- MPI_Comm comm: comunicador
- MPI_Status *status: estrutura para que depois da execução da função detalhes de transmissão possam ser inspecionados

(16) Atividade

(1) Escreva um programa em C com MPI que envia uma frase (string) de um processo origem para um processo destino.

(2) Implemente um programa similar ao anterior, em que o remetente faz uma pergunta para o destinatário que responde com a frase "Pergunta lá no posto Ipitanga!".