AULA 9

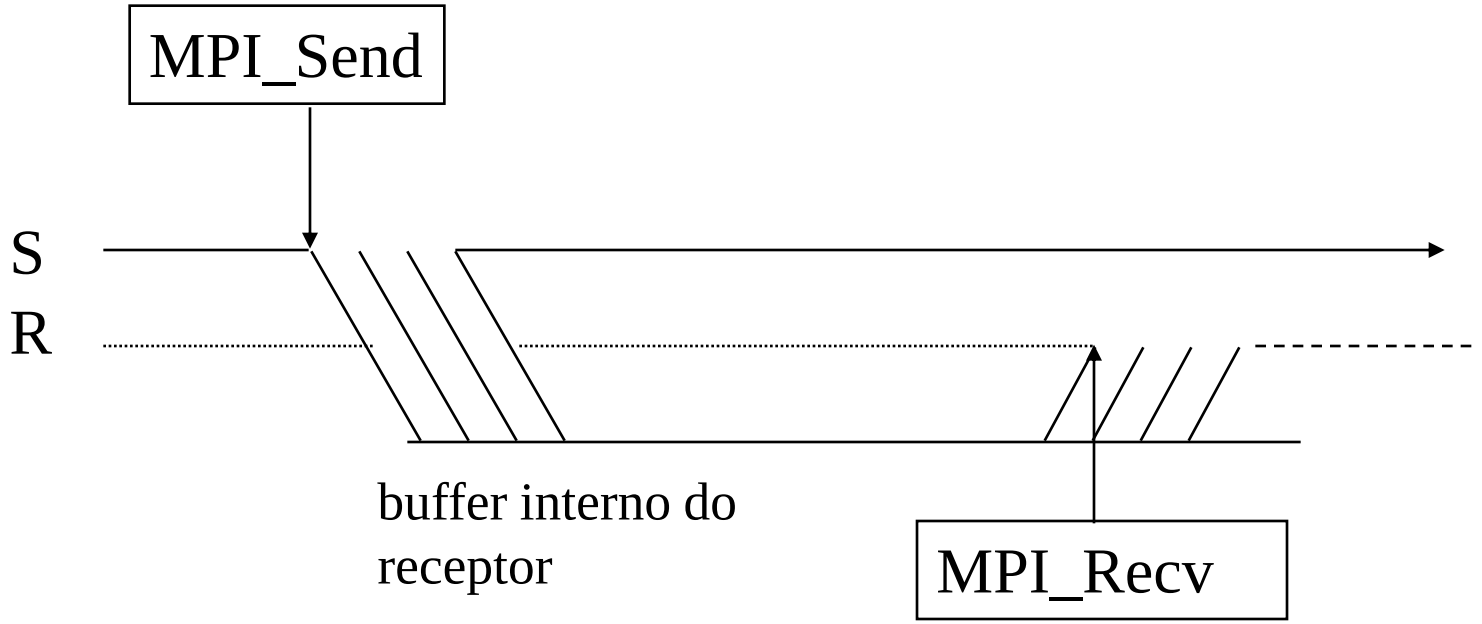
Programação Paralela e Distribuída

Comunicação: comportamento bloqueante

Existem 4 modos de comunicação bloqueante.O modo de comunicação é selecionado pela escolha da rotina send. A rotina receive a ser utilizada pode ser bloqueante ou não bloqueante.

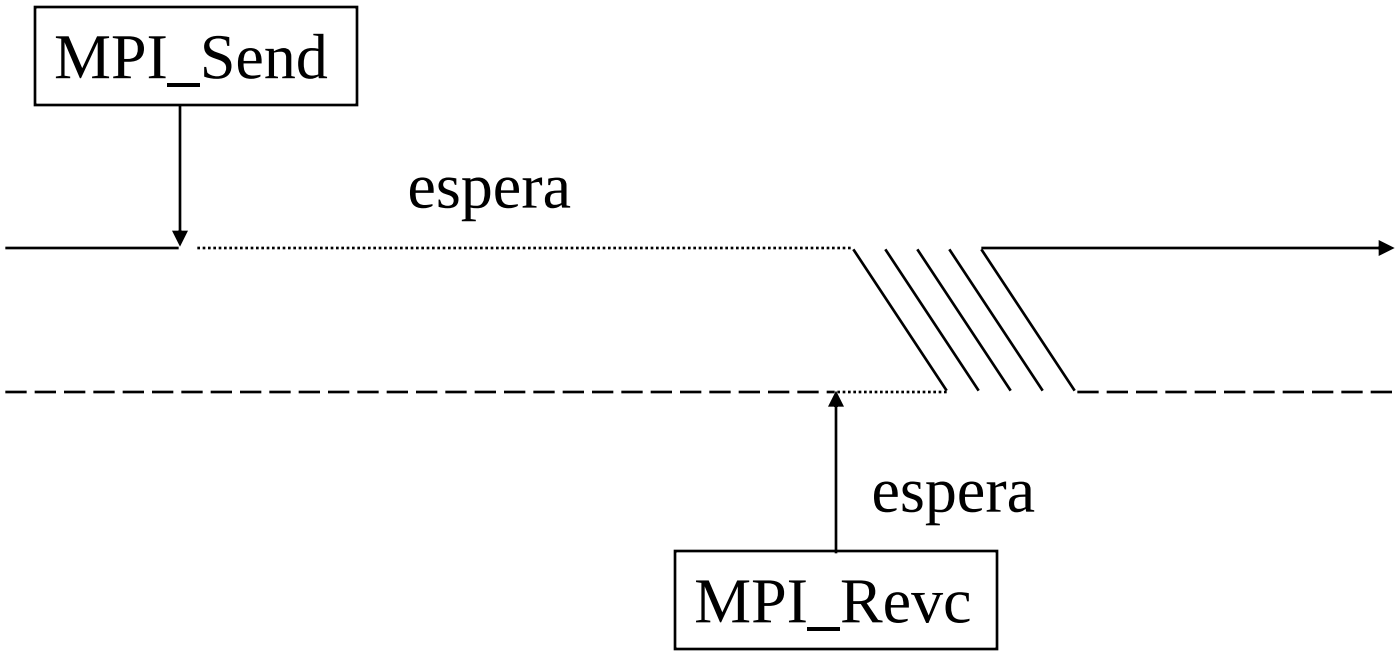
Envio Padrão Bloqueante

- O comportamento do sistema depende do tamanho da mensagem, se é menor ou igual ou maior do que um limite. Este limite é definido pela implementação do sistema e do número de tarefas na aplicação.
- mensagem<= limite



Envio Padrão Bloqueante

- mensagem>limite



Programação: modelo mestre-escravo em SPMD

```
#include <mpi.h>
#define WORKTAG   1
#define DIETAG        2
#define NUM_WORK_REQS       10
static  void master();
static  void slave();
void  (*tab_func[10])( );
void func0()
  { int result=0;
    MPI_Send(&result,1,MPI_INT,0,0,
                    MPI_COMM_WORLD);
}
...
void funcn()
   { ........}
```

Programação: modelo mestre-escravo em SPMD

```
int main(int argc, char *argv[])
{ int myrank;
   MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
if (myrank==0) {master();}
 else { slave();}
 MPI_Finalize();
 return(0);
}
```

```
/*master: envia pedidos de trabalhos aos escravos e coleta resultados*/
void master()
{ int ntasks,rank,work;
   int result;
   MPI_Status status;
  MPI_Comm_size(MPI_COMM_WORLD,&ntasks);
  work=NUM_WORK_REQS-1;
  tab_func[0]=func0;
  tab_func[2]=func1;
  .....
  tab_func[work-1]=funcn;
  for (rank=1;rank<ntasks;++rank){
     MPI_Send(&work,1,MPI_INT,rank,WORKTAG,
                    MPI_COMM_WORLD);
     work--;
   }
```

```
/*recebe um resultado de qualquer escravo e atribue um novo
    trabalho até execução de todos*/
while (work>0) {

    MPI_Recv(&result,1,MPI_INT,MPI_ANY_SOURCE,MP
    I_ANY_TAG,MPI_COMM_WORLD,&status);
     MPI_Send(&work,1,MPI_INT,status.MPI_SOURCE,
                        WORKTAG,MPI_COMM_WORLD);
      work--;
   }
```

```
/* Recebe resultados dos escravos sem atribuir novos
    trabalhos*/
for (rank=1;rank<ntasks;rank++); {

    MPI_Recv(&result,1,MPI_INT,MPI_ANY_SOURCE,MP
    I_ANY_TAG,MPI_COMM_WORLD,&status);
/* comunica aos escravos para exit */
    MPI_Send(0,0,MPI_INT,status.MPI_SOURCE,DIETAG,
                    MPI_COMM_WORLD);
  }

}
```

```
void slave()    /*slave */
{  int result;
   int       work;
   MPI_Status      status;
   tab_func[0]=func0;
   tab_func[2]=func1;
    .....
   tab_func[work-1]=funcn;
   for(;;) {
     MPI_Recv(&work,1,MPI_INT,0,MPI_ANY_TAG,
                      MPI_COMM_WORLD,&status);
     if (status.MPI_TAG==DIETAG)
           break;
     (*tab_func[work])();
}
}
```

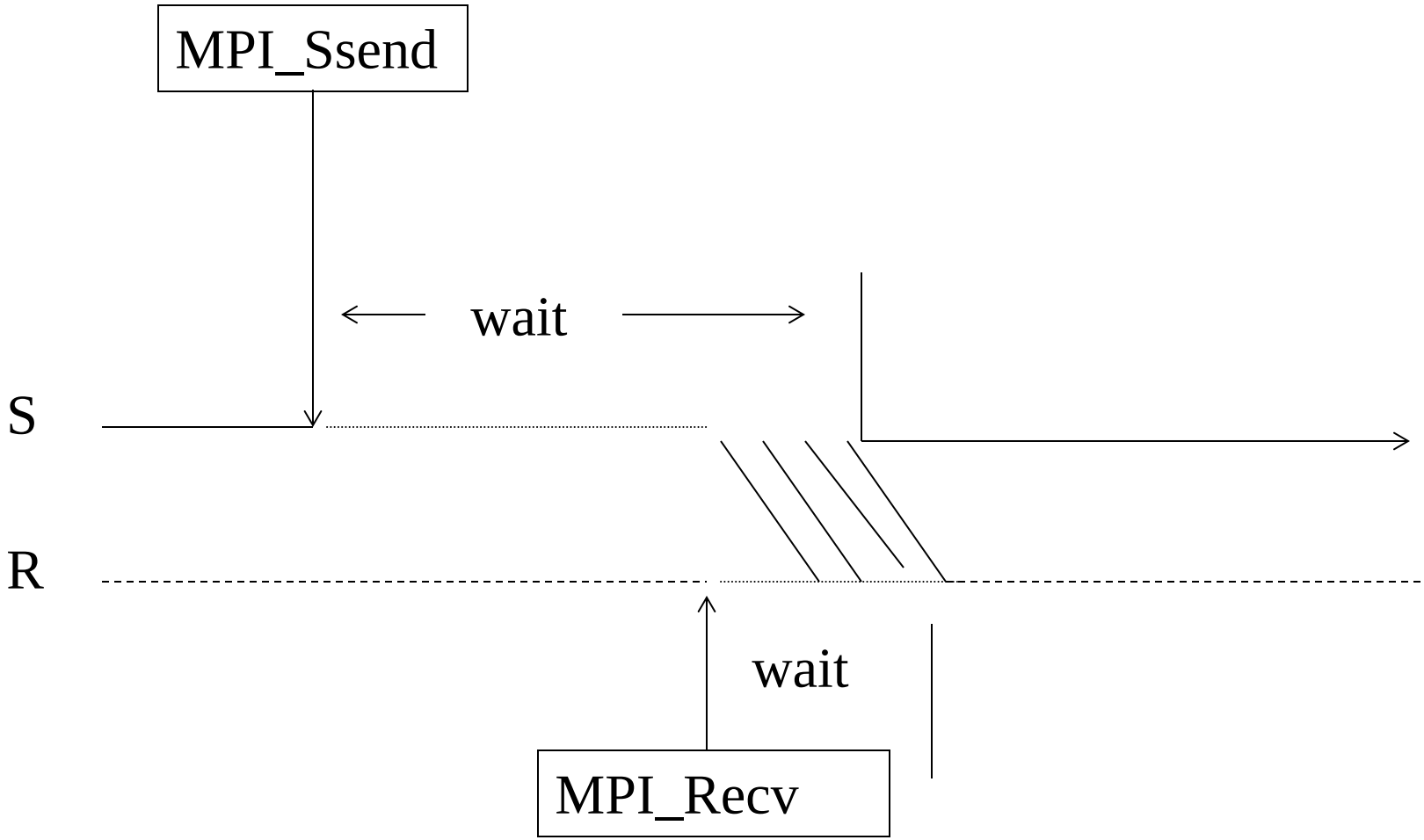
Exercício para casa

- Implementar aplicando o esquema mestre-escravo:
  - mestre: atribui a cada escravo a função a ser executada . Recebe cada resultado retornado pelo escravo e imprime. Total de funções=10.
  - escravo: executa a função atribuída pelo mestre, retorna o resultado ao mestre e solicita nova função. Finaliza ao receber DIETAG.

Blocking Syncronous Send

- MPI_Ssend(....)
- A tarefa transmissora envia para a tarefa receptora uma mensagem “ready to send”. Quando a tarefa receptora executa uma chamada receive, ela envia para a tarefa transmissora uma mensagem de “ready to receive”. Os dados são então transferidos.

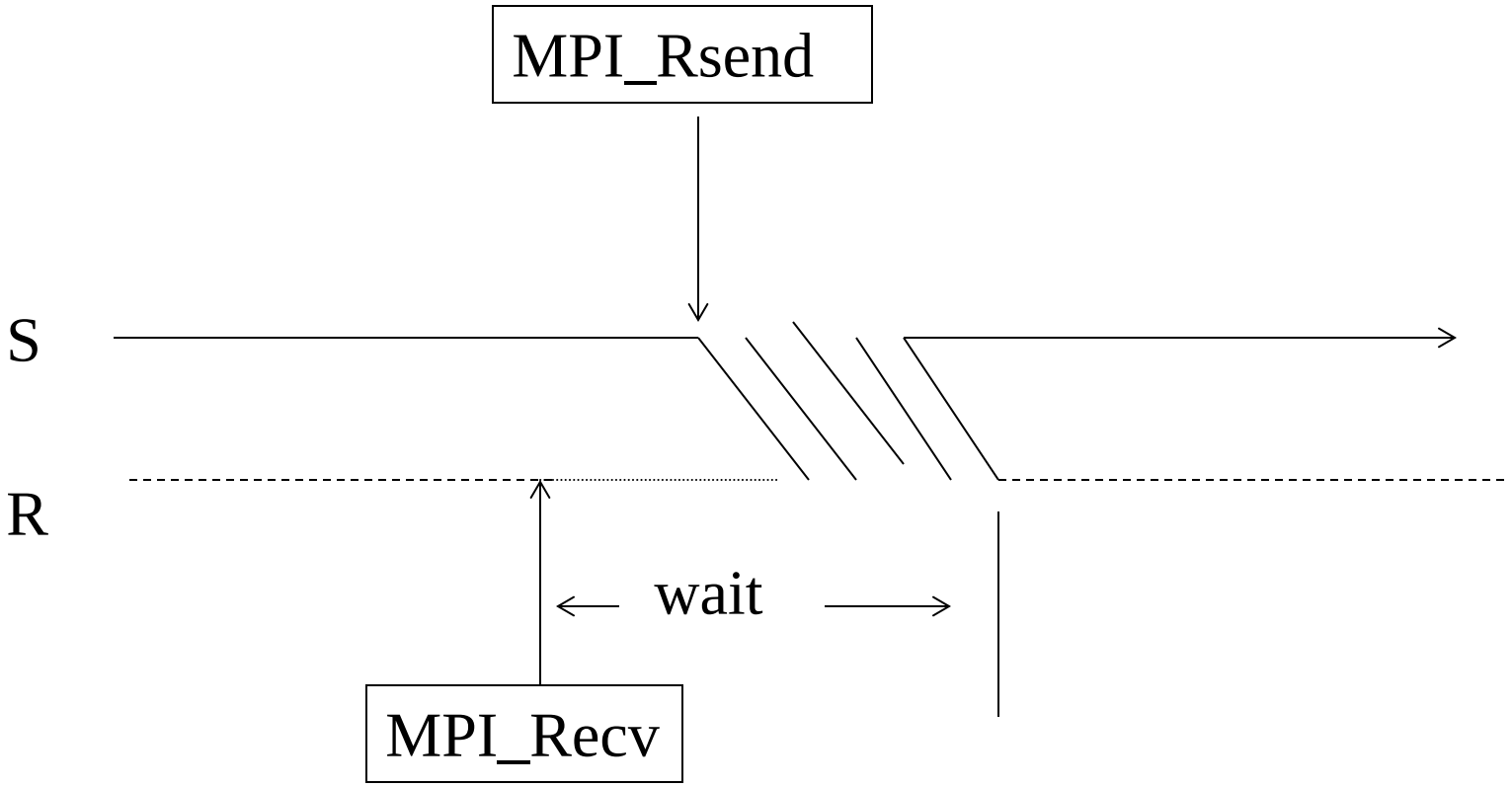
Blocking Syncronous Send



Blocking Ready Send

- MPI_Rsend (...)
- O modo “ready” envia a mensagem na rede. Requer que uma notificação de “ready to receive”tenha chegado. Se esta notificação não chegou ainda, causará um erro. Por default a execução será finalizada. O programador pode associar um tratador de erro.

Blocking Ready Send



Blocking Buffered Send

- No transmissor o dado é copiado do buffer de mensagem para um buffer do usuário, e então a execução retorna. O dado será copiado do buffer do usuário sobre a rede quando uma notificação de “ready to receive”tiver chegado.
- Gerenciamento do buffer

O usuário deve prover o buffer: pode ser um array alocado estaticamente ou dinamicamente com malloc. A quantidade de memória deve exceder a soma dos dados da mensagem, como os headers que devem também ser armazenados.

Blocking Buffered Send

- Este espaço alocado deve ser identificado como um buffer do usuário:

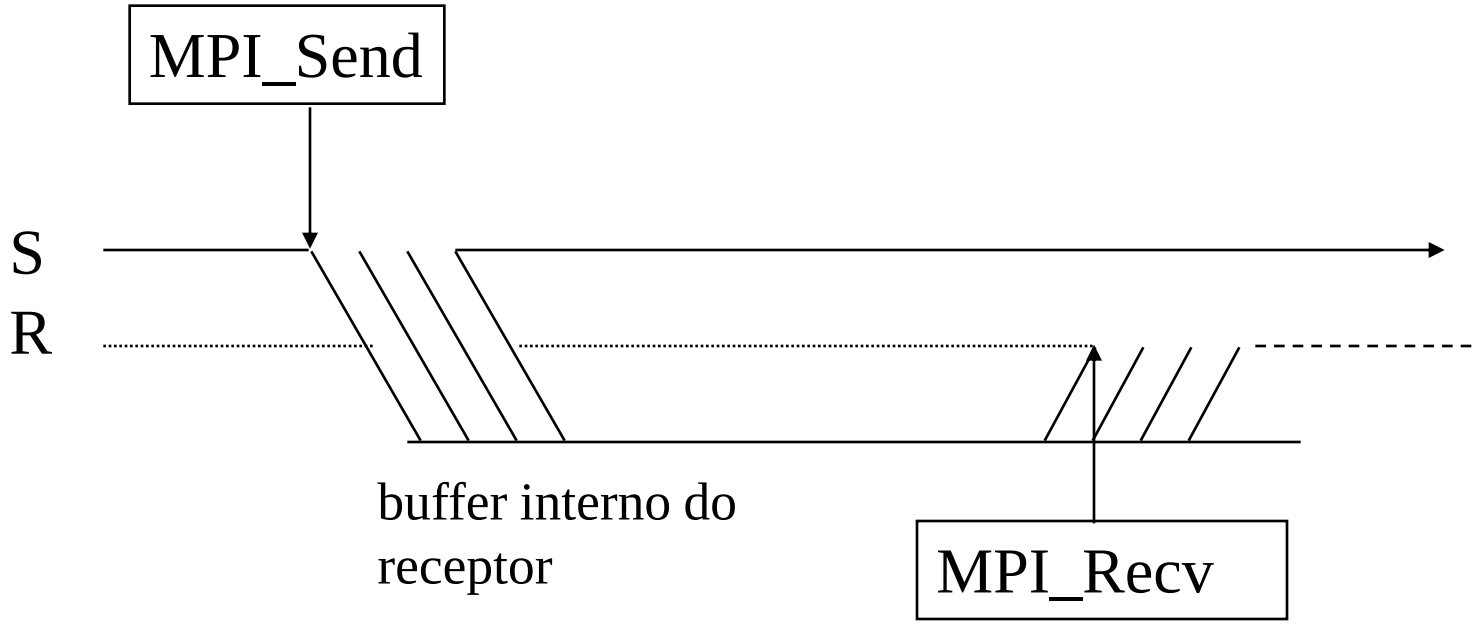
  MPI_Buffer_attach(...)

  MPI_Buffer_detach(...)

Exemplo: mpi_somaBSend.c

Envio Padrão Bloqueante

- O comportamento do sistema depende do tamanho da mensagem, se é menor ou igual ou maior do que um limite. Este limite é definido pela implementação do sistema e do número de tarefas na aplicação.
- mensagem<= limite



exercício

Implementar aplicando pipeline e MPI_Bsend para envio:

Versão mpi : tarefa 1 (rank 0) tarefa2 (rank 1) tarefa3 (rank 2) tarefa4 (rank 3)

tarefa1: gera matriz X tarefa2: gera matriz Y tarefa2:gera matriz Y

tarefa3: gera matriz Z tarefa4: W=X+Y+Z .

Tarefa 1:

- Inicializa matriz X[500][500]
- gera elemento da linha i da matriz X[500][500]
- Envia linha gerada para rank 3 (tarefa4)

Tarefa 2:

- Inicializa matriz Y[500][500]
- gera elemento da linha i da matriz Y[500][500]
- Envia linha gerada para rank 3 (tarefa4)

Exercicio (continuação)

Tarefa 3:

- Inicializa matriz Z[500][500]
- gera elemento da linha i da matriz Z[500][500]
- Envia linha gerada para rank 3 (tarefa4)

Tarefa 4

- Para i de 0 a 500:

  recebe linhas i geradas e enviadas por tarefa1, tarefa 2 e tarefa 3, calculando cada elemento da matriz W (W = X + Y + Z)

OBS: o arquivo mpi-pipe-seq.c é uma versão sequencial do problema.