

The background of the slide features a light blue gradient with a faint, semi-transparent image of classical architectural columns on the left side. The columns are white with detailed capitals and fluted shafts, set against a darker blue background.

# Computação paralela e distribuída

Computação paralela com a utilização de bibliotecas de passagem de mensagem  
(*Message Passing Computing*)

Prof. Dr. Luciano José Senger

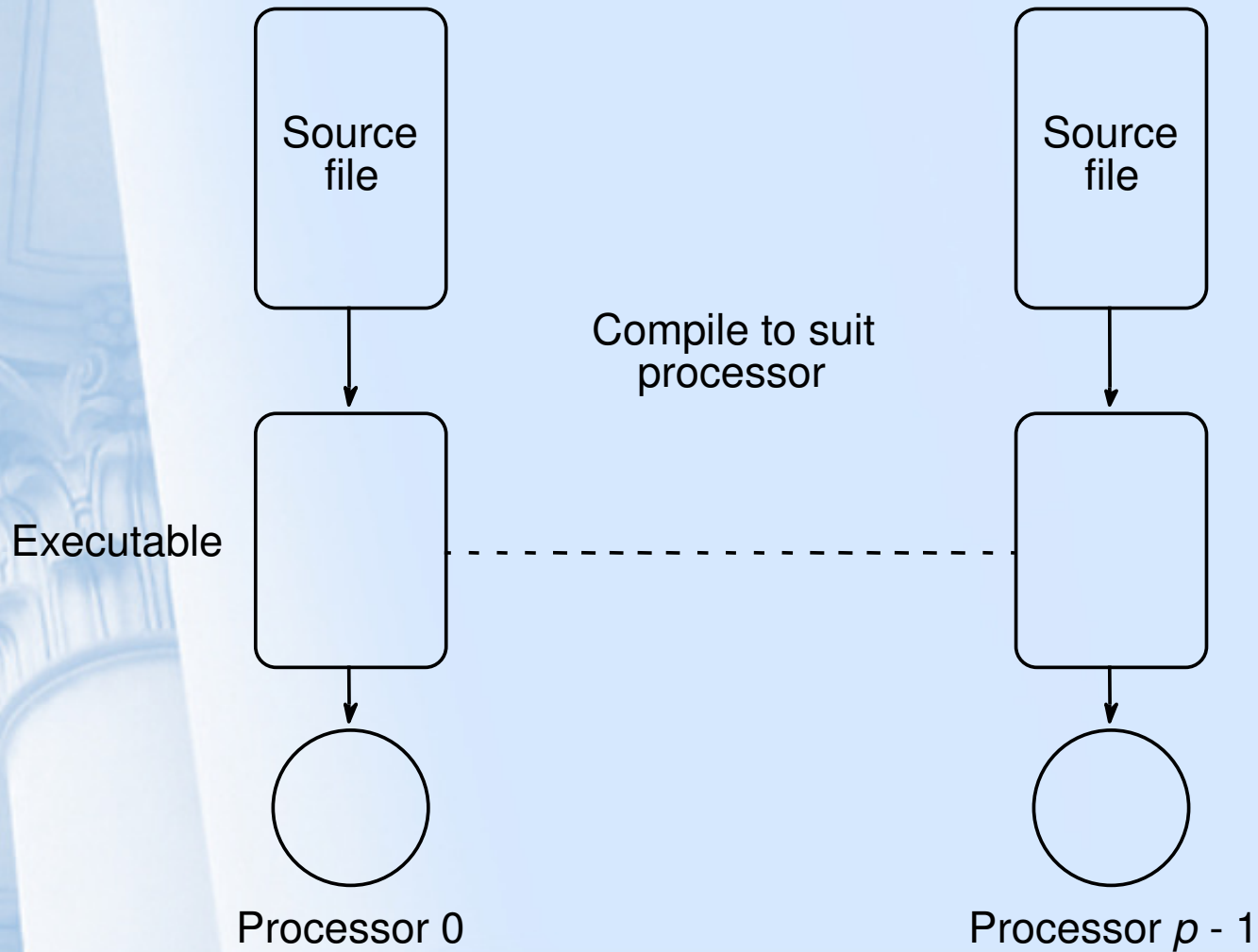
# Introdução

- Computação paralela
  - Conjunto de tarefas (processos) que interagem para resolver um determinado problema
  - A interação entre processos pode ser realizada através da memória, se a arquitetura for de memória compartilhada
- Troca de mensagens
  - Permite que um processo tenha acesso ao conteúdo da memória de outro processo
  - Troca de informações e sincronização
- Computação com troca de mensagens
  - Necessita de dois mecanismos básicos:
    - Método de criação de processos em diferentes computadores e controlar a execução
    - Mecanismos para troca de mensagens
      - Ponto a ponto
      - Coletivas

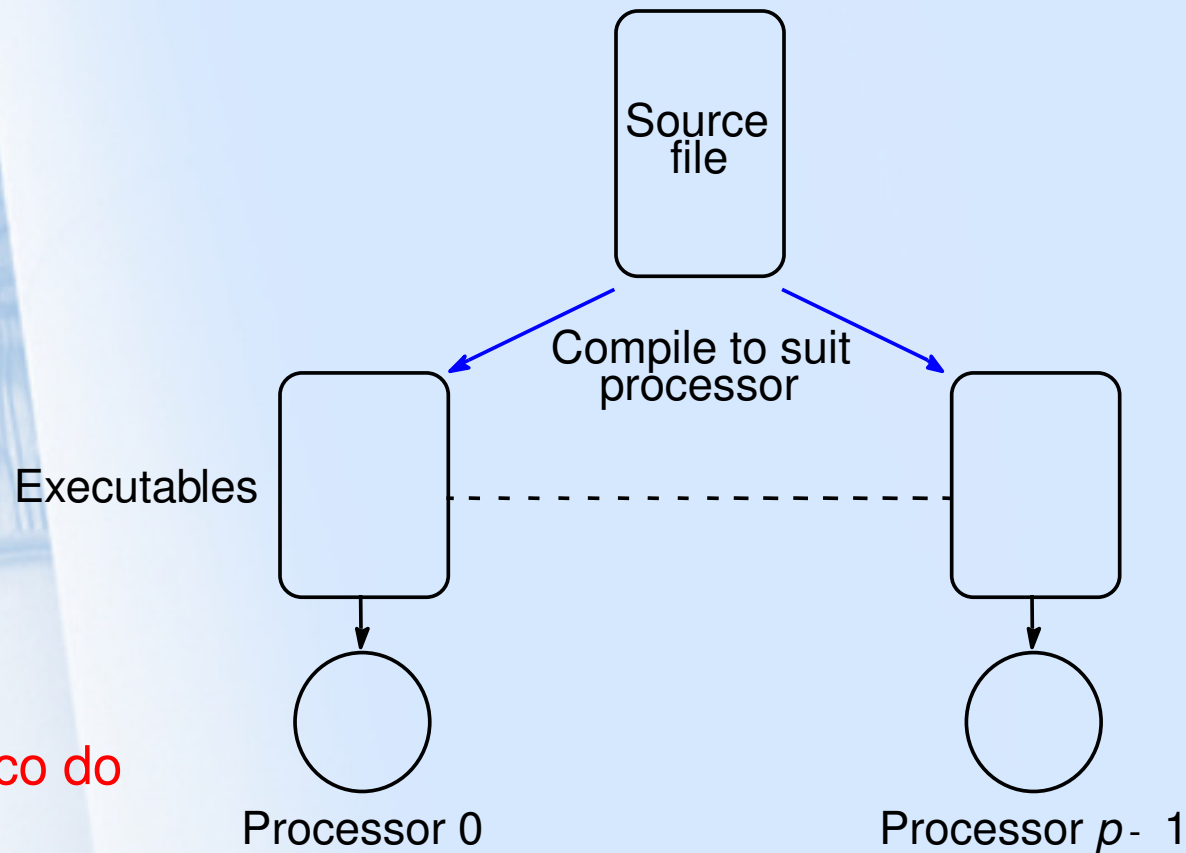
# Modelos de computação

- MPMD Multiple Program, Multiple Data
  - Cada elemento de processamento executa um programa diferente
- SPMD Single Program, Multiple Data
  - Cada elemento de processamento roda o mesmo programa
  - Cada elemento de processamento trabalha com dados diferentes
  - A variação da computação em elementos de processamento diferentes é realizada através de comandos de desvio de execução (*if, switch*)

# Multiple Program Multiple Data (MPMD)

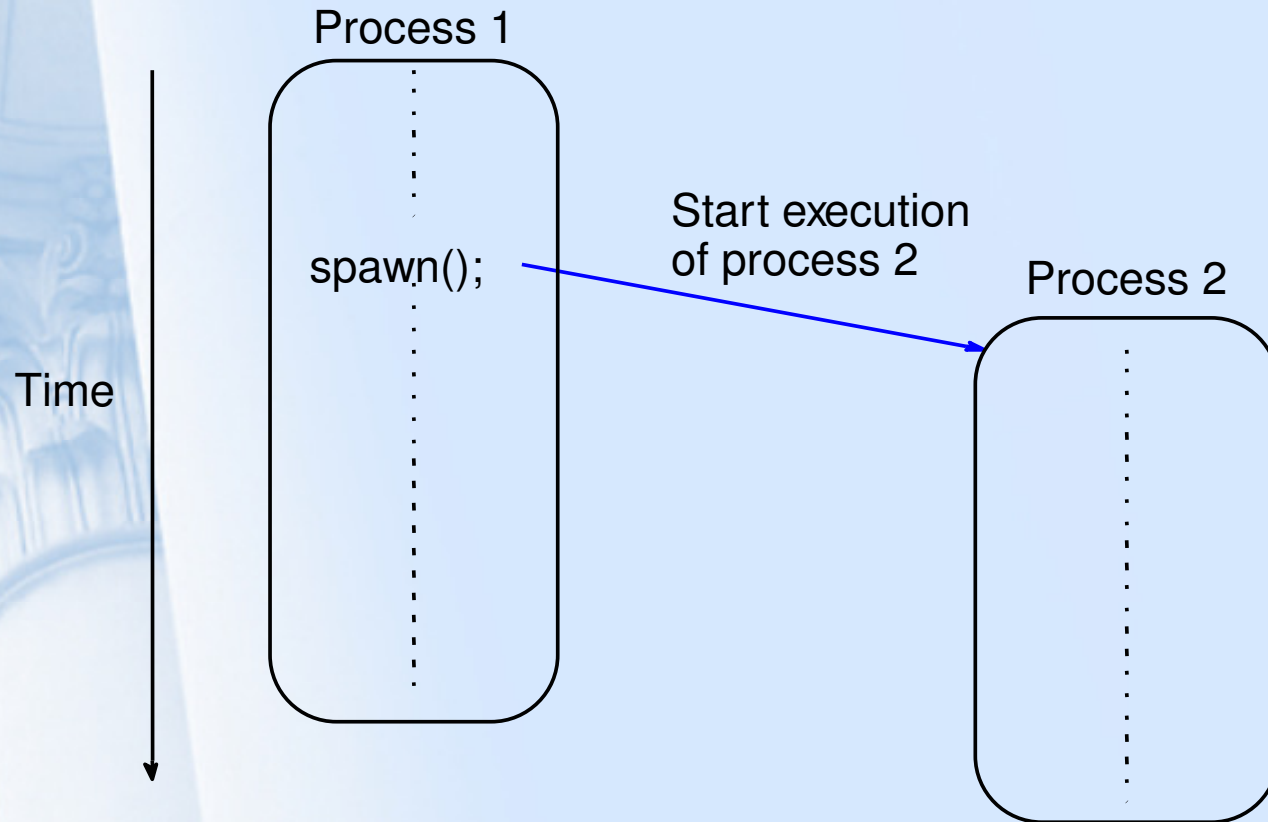


# Single Program Multiple Data (SPMD)



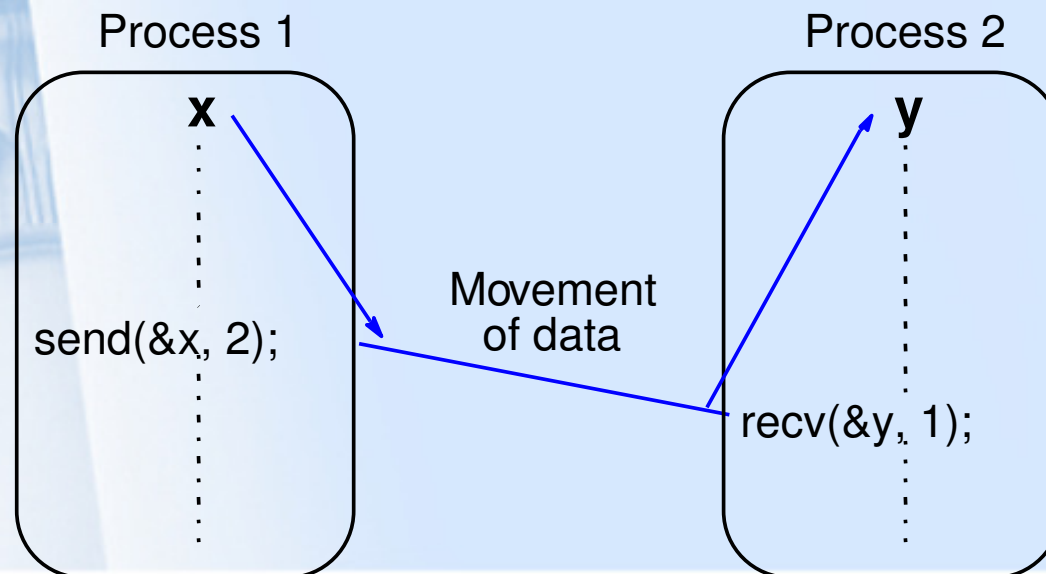
Modelo básico do  
MPI

# Multiple Program Multiple Data (MPMD)



# Rotinas de comunicação ponto a ponto

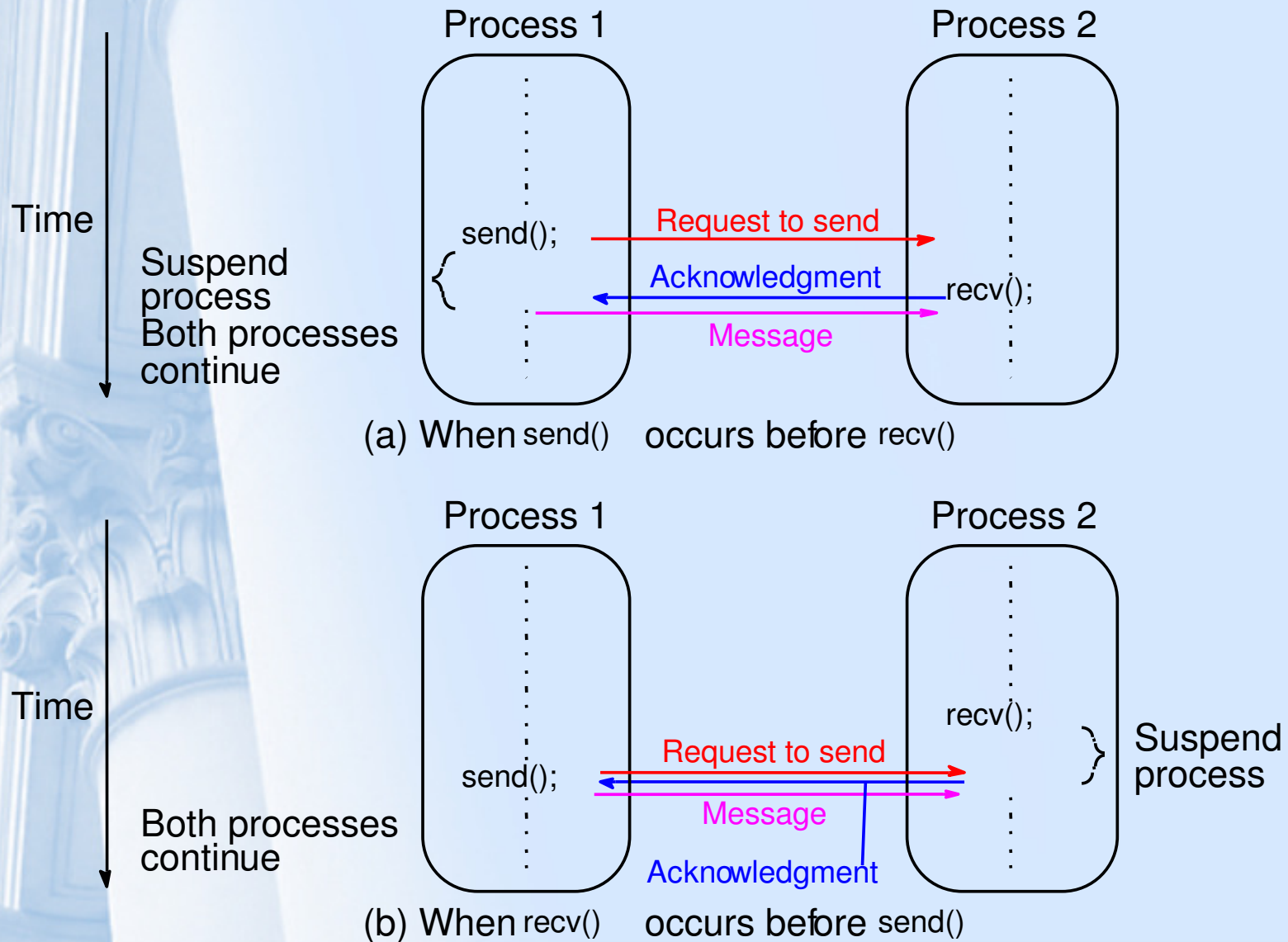
- Rotinas básicas de comunicação
  - Necessitam de um identificador da tarefa e do buffer a ser enviado (informações adicionais são necessárias)
  - Buffer geralmente é uma sequência de Bytes
  - Biblioteca deve garantir o empacotamento da mensagem e sua entrega
- Tipos de comunicação
  - Síncrona
  - Assíncrona
  - Bloqueante
  - Não-bloqueante



## Comunicação síncrona

- Retornam o controle ao processo quando a transferência da mensagem é realizada de fato
- Realizam duas ações
  - Transferência de dados através do envio da mensagem
  - Sincronização entre processos, por garantir que os processos fiquem bloqueados até que a transferência seja realizada

# Comunicação síncrona com send() e recv() usando um protocolo de 3 partes (three-way)

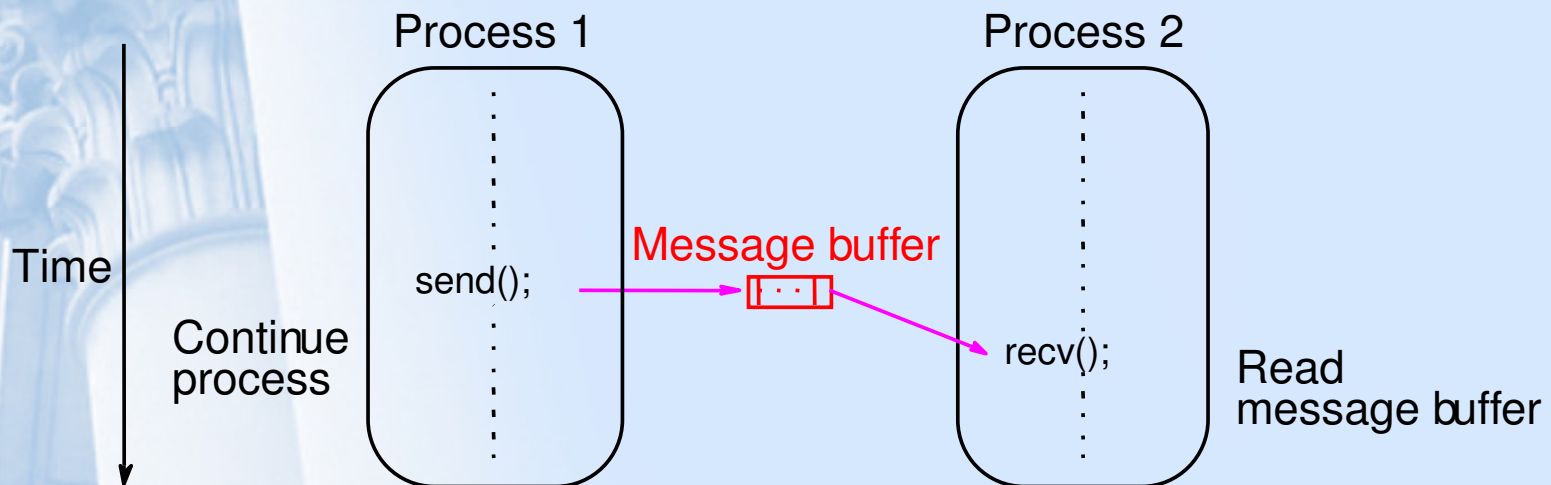


# Comunicação assíncrona

- Rotina de comunicação que não espera que a comunicação seja finalizada para retornar
  - Necessita de armazenamento local da mensagem (bloqueante)
- Não permite uma sincronização real entre os processos, mas permite que os processos sejam liberados rapidamente para seguir com a execução (no caso de um *send()*)
- Rotinas bloqueantes e não bloqueantes
  - Rotinas bloqueantes retornam após as ações locais serem realizadas
  - Rotinas não bloqueantes retornam imediatamente
    - Assumem que os dados usados para a transferência não serão modificados subsequentemente pelo programa.
    - Programador deve garantir consistência

# Comunicação assíncrona e sua implementação com buffers de mensagens

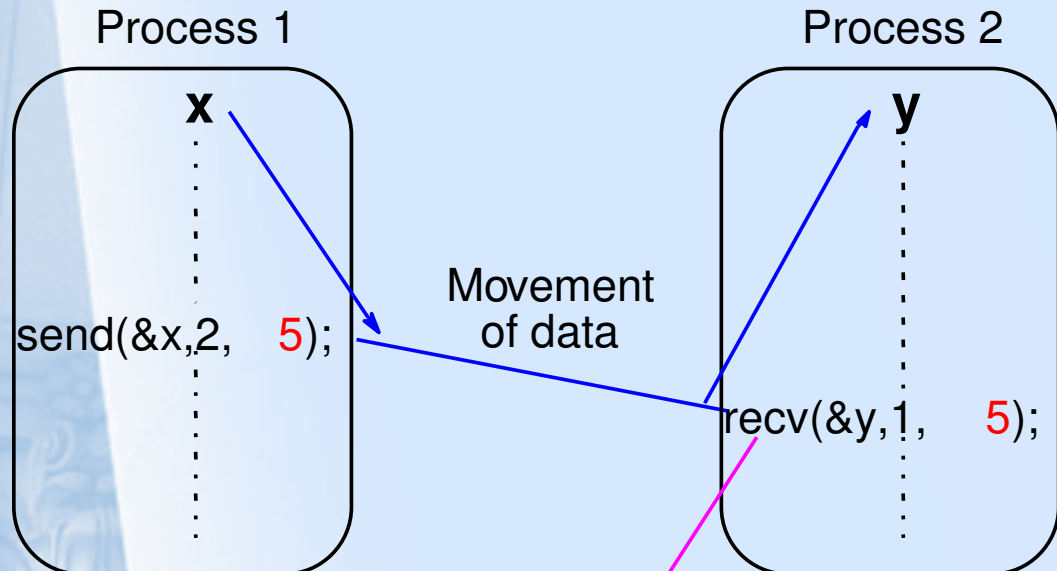
- Vantagem
  - Desempenho melhorado na computação/comunicação
- Desvantagem
  - Cópia de dados para o buffer
  - Rotina assíncrona pode tornar-se síncrona, no caso de falta de espaço em buffers



## Uso de rótulo (envelope) em mensagens

- Cada mensagem é empacotada com um campo de rótulo (envelope)
  - Identificador inteiro
  - Usado para diferenciar as mensagens que estão sendo enviadas
- Se não for necessária a utilização de um campo de tag, *wildcards* (p.e. \* ) podem ser empregados para especificar que a rotina de *recv()* aceita qualquer mensagem (qualquer *send()*)

# Exemplo de utilização de tag (tag=5)

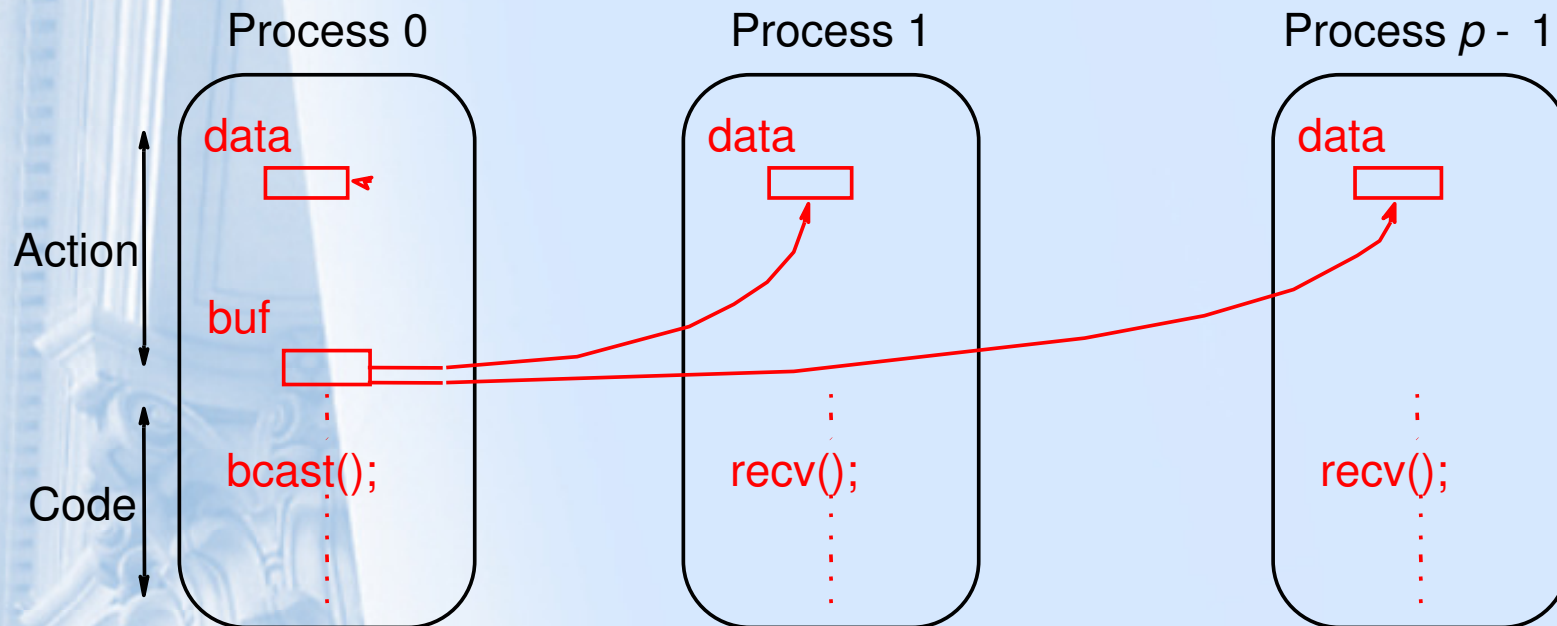


Waits for a message from process 1 with a tag of 5

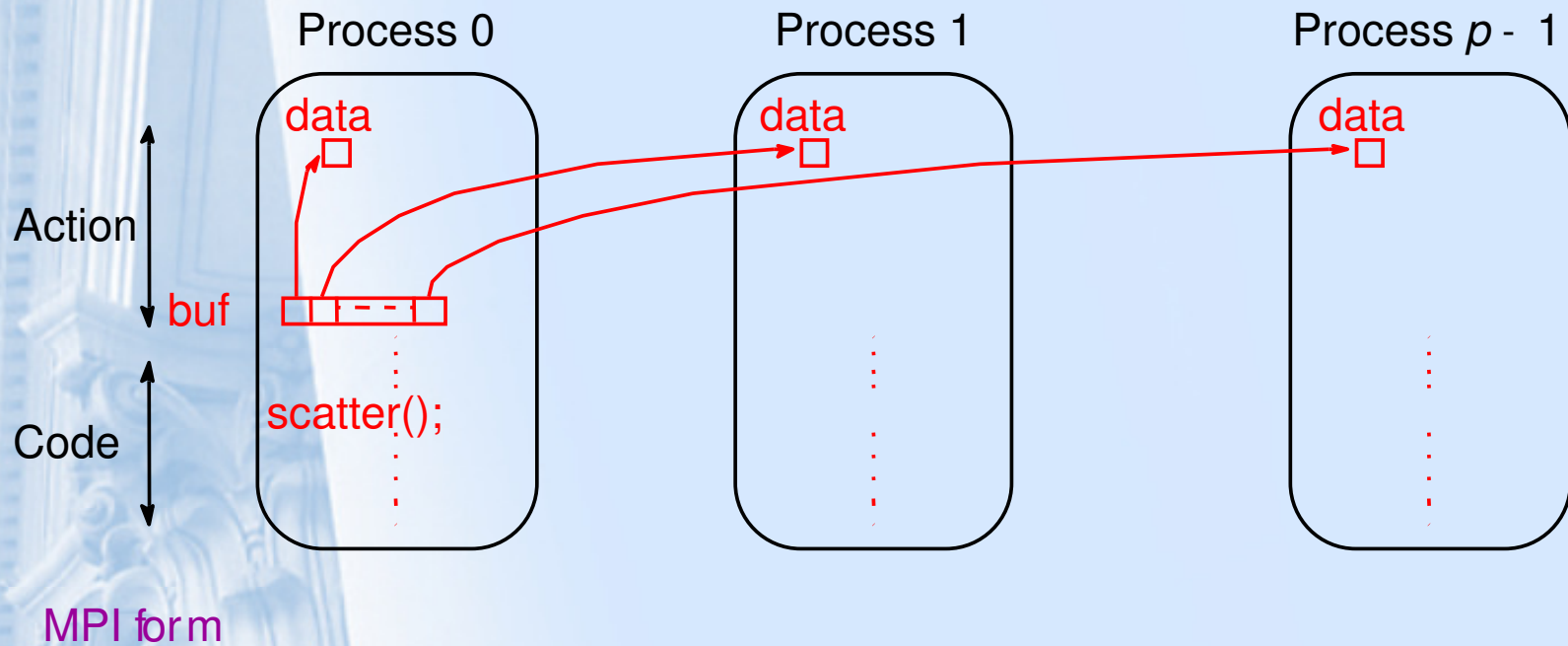
# Comunicação coletiva

- Permite que a comunicação seja realizada através do envio/recebimento de mensagens de/para um grupo de processos
- Exemplos de rotinas de envio de mensagem para com conjunto de receptores
  - Broadcast – envio de uma mesma mensagem para todos os processos da aplicação
  - Multicast – envio de uma mensagem para um grupo definido dos processos da aplicação
  - Scatter – envia elementos de um buffer de forma que o  $i$ -ésimo elemento do buffer é enviado para o  $i$ -ésimo processo
- Exemplos de rotinas para recebimento de mensagens dentro de um grupo
  - Gather – recebe elementos de um buffer, de forma que cada receptor envia uma parte do buffer
  - Reduce – recebe mensagens de vários receptores e realiza uma operação com os dados recebimentos (p.e. Min, Max, Soma, etc.)

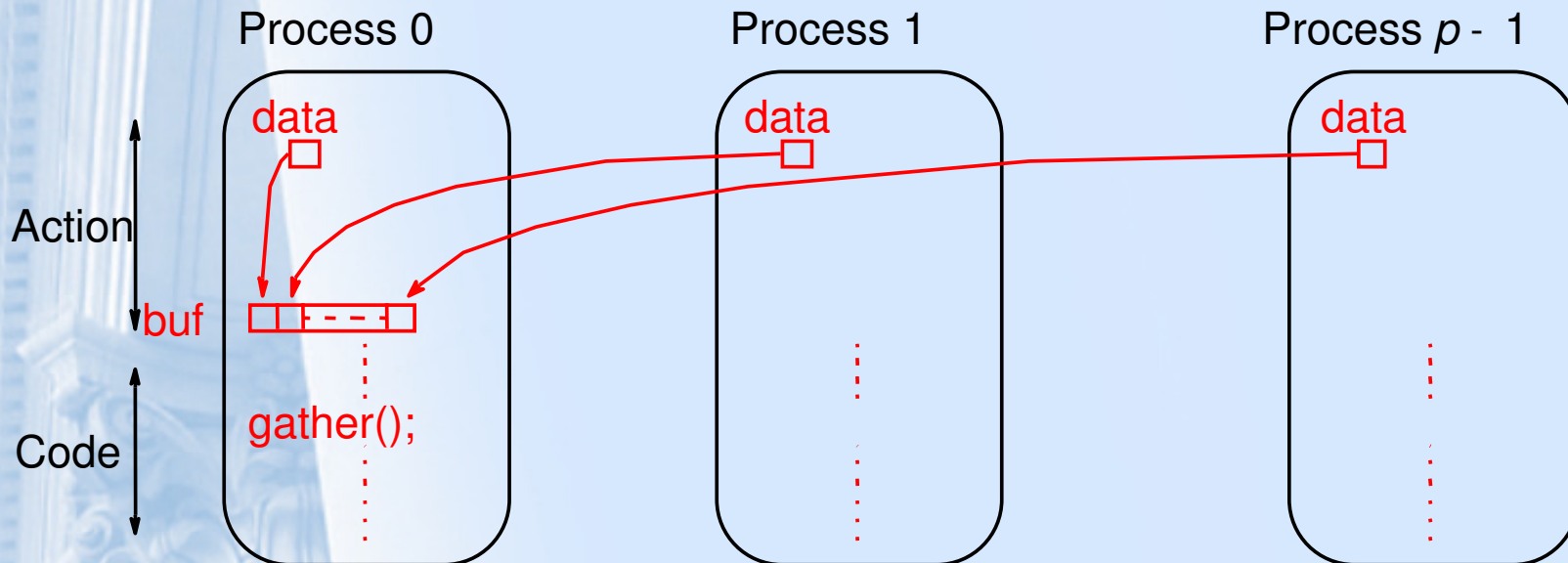
# Comunicação coletiva com o uso de rotina de broadcast



# Comunicação coletiva com o uso de Scatter

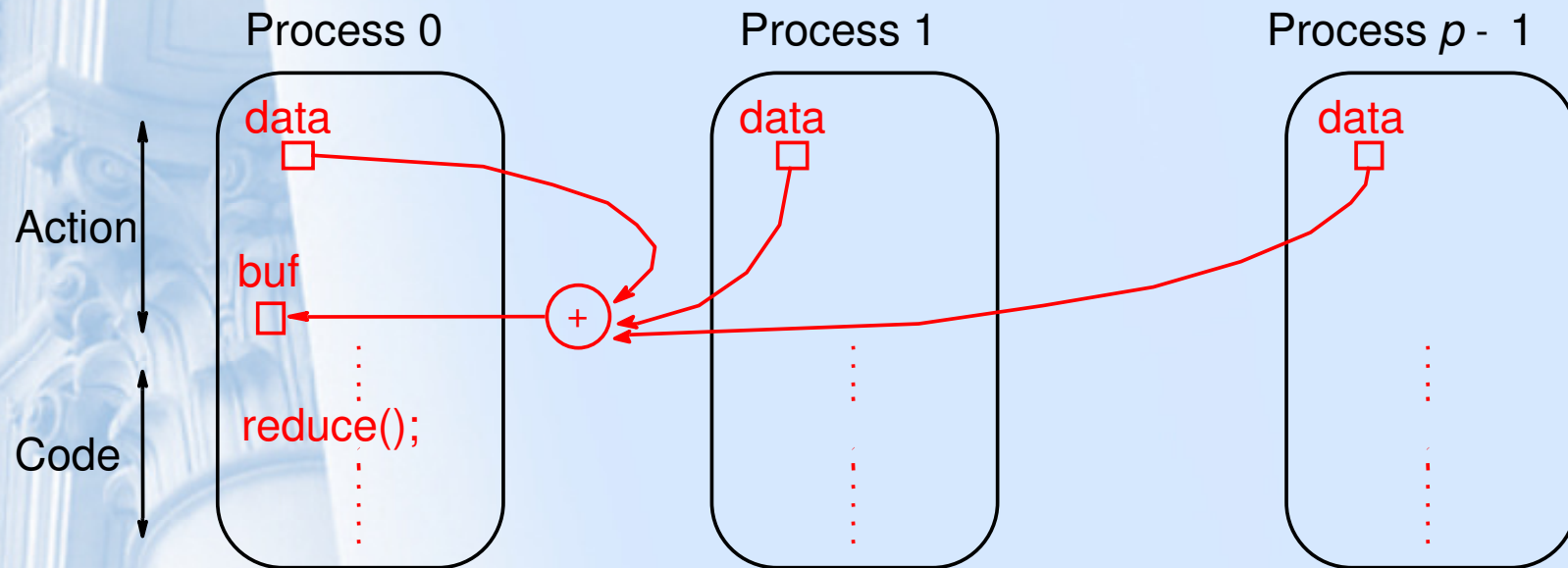


# Comunicação coletiva com o uso de gather



# Comunicação coletiva com o uso de Reduce

- Rotina **Reduce** é similar ao uso de **gather** combinado com uma operação aritmética
- Exemplo:
  - Processo 0 recebe mensagens de vários processos e realiza a soma dos conteúdos das mensagens



# Discussão

- Bibliotecas de passagem de mensagem
  - Adicionam mecanismos para permitir a execução paralela em linguagens imperativas
  - Rotinas básicas para comunicação e sincronização
- Exemplos
  - PVM (parallel virtual machine)
  - MPI (message passing interface)
- Formas de comunicação
  - Uso depende da aplicação
  - Outras formas e mecanismos
    - mensagens com manipulador associado
      - Registra-se um tag para um manipulador
      - Função ou procedimento é invocado automaticamente ao processo receber a mensagem
    - Contexto de mensagens