




Processamento Paralelo e Distribuído

Marcelo Trindade Rebonatto

Formas de Comunicação


Processamento Paralelo e Distribuído - Prof.: *Marcelo TrindadeRebonatto*



Roteiro

- Trocas de Mensagens
 - Conceitos Gerais
 - Primitivas básicas e adicionais
- Modelos de comunicação
 - Síncrona
 - Assíncrona
 - RPC

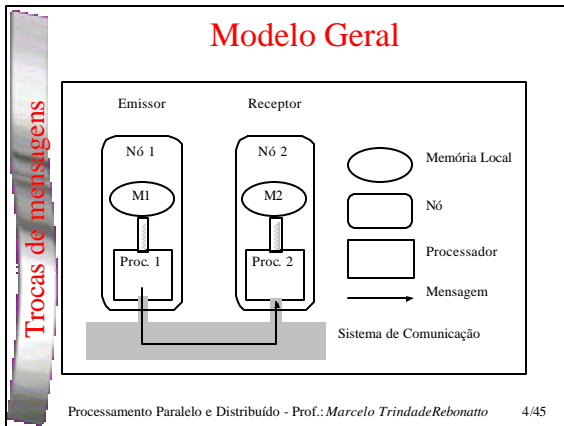
Processamento Paralelo e Distribuído - Prof.: *Marcelo TrindadeRebonatto* 2/45



Contexto

- Diversos processadores executando uma tarefa
 - Processo(s) executado(s)
 - Comunicação e sincronização
- Memória distribuída entre os processadores
 - Impossível o uso de técnicas de variáveis compartilhadas
 - Par processador + memória = nó
 - Nós interligados através de uma rede de comunicação

Processamento Paralelo e Distribuído - Prof.: *Marcelo TrindadeRebonatto* 3/45



Trocas de mensagens

Interface de Comunicação

- Comunicação entre dois processos
 - Emissor
 - Receptor
- Trocas de mensagens com sincronização implícita
 - Uso da rede de comunicação
 - ☆ Canais
 - Duas primitivas básicas:
 - ☆ SEND
 - ☆ RECEIVE

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 5/45

Trocas de mensagens

Processos

- Um processo é semelhante a um programa seqüencial
 - Utilização dos conceitos da programação seqüencial
 - ☆ Fluxo seqüencial dos comandos
 - ☆ Desvios condicionais
 - ☆ Repetições
- Dificuldade: determinação e manutenção do estado global
 - Sincronização

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 6/45

Trocas de mensagens

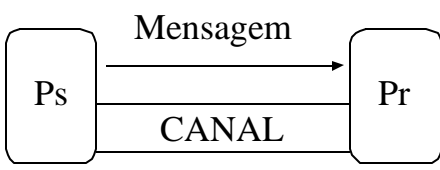
Canais

- Únicos objetos compartilhados entre processos
- Compartilhados por 2 ou mais processos
- Abstração da ligação física
 - Sistema de Comunicação
- Sincronização
 - 1 mensagem não pode ser recebida enquanto não for enviada

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto
7/45

Trocas de mensagens

Canais



- Ps: Processo emissor
 - SEND
- Pr: Processo receptor
 - RECEIVE

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto
8/45

Trocas de mensagens

Primitivas básicas - SEND

- Envia uma mensagem de um processo para outro
- Forma geral:

`SEND(message, messagesize, target, type, flag)`
- Forma MPI:

`MPI_Send(message, count, type, target, tag, communicator)`

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto
9/45

Primitivas básicas - SEND

- `message`: contém os dados a serem enviados
- `messagesize`: tamanho da mensagem em bytes
- `target`: especifica o receptor
 - Não necessita ser um único
- `type`: tipo dos dados enviados definido pelo usuário
- `flag`: status do envio

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 10/45

Primitivas básicas - SEND

- **Blocking**: bloqueante
 - Paralisa a execução do processo que envia uma mensagem até que seja realizado o receive no processo destino
- **Nonblocking**: não bloqueante
 - Após o envio da mensagem, o processo que a enviou segue imediatamente seu fluxo de execução
- **Flag**
 - Pode ser utilizado para determinar se o `send` é ou não bloqueante

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 11/45

Primitivas básicas - RECEIVE

- Recebe uma mensagem de algum processo
- Forma geral:

```
RECEIVE(message, messagesize,
         source, type, flag)
```

- Forma MPI:

```
MPI_Recv(message, count, datatype,
          source, tag, communicator,
          status)
```

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 12/45

Primitivas básicas - RECEIVE

- **message**: identifica a localização onde os dados recebidos serão armazenados
- **messagesize**: número máximo de bytes a ser recebido em 'message'
- **source**: indica de qual nodo é a mensagem
 - Não necessariamente apenas 1 nodo
- **type**: indica o tipo de dados recebido na mensagem
- **flag**: status do recebimento

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 13/45

Primitivas básicas - RECEIVE

- **Blocking**: bloqueante
 - Suspende a execução do processo até a chegada de uma mensagem que possa ser lida do buffer de comunicação
- **Nonblocking**: não bloqueante
 - Retorna o controle ao programa mesmo se nenhuma mensagem puder ser lida do buffer de comunicação
- **Flag**
 - Pode ser utilizado para determinar se o **receive** é ou não bloqueante

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 14/45

Primitivas adicionais

- Identificação do processador atual e número de processadores
- Forma geral:

```
WHOAMI (processorid, numofprocessors)
```

- Forma MPI:

```
MPI_Comm_rank(communicator, me)
```

```
MPI_Comm_size(communicator, np)
```

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 15/45

Trocas de mensagens

Primitivas adicionais

- `processorid`
 - Número do processador em que o sistema está sendo executado
 - Identificação do processador atual
- `numofprocessors`
 - Número de processadores em que a aplicação está sendo executada
 - Quantidade de processadores que contém a máquina paralela

Processamento Paralelo e Distribuído - Prof.: *Marcelo TrindadeRebonatto* 16/45

Formas de Comunicação

Trocas de mensagens

- MP pode ser implementada em arquiteturas de memória compartilhada
 - Modelo único de programação
 - ✧ Máquinas de memória compartilhada
 - ✧ Máquinas de memória distribuída
- Modelos básicos
 - Assíncrona: sem bloqueio
 - Síncrona: com bloqueio
 - Remote Procedure Call: RPC
 - Rendezvous: síncrona bidirecional

Processamento Paralelo e Distribuído - Prof.: *Marcelo TrindadeRebonatto* 17/45

Formas de Comunicação

Comunicação assíncrona

- 1 canal é uma fila não limitada de mensagens
 - Contém mensagens enviadas e ainda não recebidas
- 1 processo coloca 1 nova mensagem no fim do canal com a primitiva `send`
 - não bloqueante devido canal ser ilimitado
- 1 processo retira (e acessa) a mensagem que está no início da fila com a primitiva `receive`
 - bloqueante se não há mensagens no canal

Processamento Paralelo e Distribuído - Prof.: *Marcelo TrindadeRebonatto* 18/45

Formas de Comunicação

Comunicação assíncrona

- Emissor: Envia a mensagem e prossegue seu fluxo, sem bloqueios
- Receptor
 - Se não estiver pronto para receber
 - ✧ Mensagem é armazenada em um buffer
 - Pronto para receber mensagem não enviada
 - ✧ Bloqueio temporário
 - ✧ Primitivas adicionais para evitar o bloqueio

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 19/45

Formas de Comunicação

Comunicação assíncrona

- Semáforos
 - Somente em memória compartilhada
 - P: bloqueia, espera
 - V: desbloqueia

`var
s : semáforo;`

P₀

P(s)

P₁

V(s)

Libera

←

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 20/45

Formas de Comunicação

Comunicação assíncrona

- Equivalência com semáforos
 - Send equivale a primitiva V
 - ✧ Não bloqueante
 - ✧ Incrementa contador (fila)
 - Receive equivale a primitiva P
 - ✧ Eventualmente bloqueante: fila vazia ou contador em zero
 - ✧ Decrementa contador (fila)
- Quantidade de mensagens na file equivale ao contador do semáforo

Processamento Paralelo e Distribuído - Prof.: Marcelo Trindade Rebonatto 21/45

7

Formas de Comunicação

Comunicação assíncrona

- Receive
 - Mensagem é retirada do canal
 - Mensagens em um destino
 - ☆ Mesma origem: igual ordem do envio
 - ☆ Diferente origem: não há ordem global
- Exclusão mútua
 - p. ex. 2 clientes (emissores) não podem se comunicar ao mesmo tempo com o servidor
 - responsabilidade do programador
- Dead-lock: pode ocorrer se houver receive em um canal sem nenhum send

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 22/45

Formas de Comunicação

Comunicação síncrona

- Um problema da comunicação assíncrona
 - Processo que envia uma mensagem quer saber se a mesma já foi recebida
 - ☆ Receptor deve enviar outra mensagem (ack) sinalizando seu recebimento
- **Comunicação síncrona**
 - Emissor e receptor ficam *sincronizados*

```

graph LR
    Ps[Ps] -- "Mensagem" --> Pr[Pr]
    Pr -- "Acknowledgement (ack)" --> Ps
  
```

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 23/45

Formas de Comunicação

Comunicação síncrona

- Emissor
 - Envia a mensagem e aguarda até que o receptor sinalize o recebimento da mesma
 - Caso receptor não pronto
 - ☆ Transmissor bloqueado temporariamente
- Receptor
 - Quando tenta receber uma mensagem, fica bloqueado até consegui-la
- Não exige presença de buffer de comunicação

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 24/45

Formas de Comunicação

Comunicação síncrona

- Quando ambos prontos
 - Emissor pronto para enviar
 - Receptor pronto para receber
 - o emissor envia a mensagem e o receptor envia sinal confirmando recebimento
 - Ambos são desbloqueados e podem seguir seu processamento
- Dead-lock
 - Dois processos querem comunicar-se e utilizam as mesmas primitivas
 - 2 em send ou receive

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 25/45

Formas de Comunicação

Síncrona X Assíncrona

- Relação entre velocidade dos processos
 - Assíncrona
 - ☆ Processos emissores com velocidade independente dos receptores
 - Síncrona
 - ☆ Velocidade do processo mais lento regula a velocidade dos demais
- Buferização
 - Assíncrona
 - ☆ Automática (ponto de vista do usuário)
 - Síncrona
 - ☆ Complica a programação, se necessária

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 26/45

Formas de Comunicação

Síncrona X Assíncrona

- Dead-lock
 - Assíncrona
 - ☆ Processos simétricos
 - ☆ Todos os processos devem primeiro enviar e após receber as mensagens
 - Síncrona
 - ☆ Assimetria
 - ☆ Deve-se estabelecer um protocolo
 - ☆ Enquanto um processo envia, outro deve estar recebendo
 - ☆ Enquanto um processo recebe, outro deve estar enviando

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto 27/45

Formas de Comunicação

Remote Procedure Call

- Motivação

 - Algoritmos Cliente-servidor: bidirecional

 - ☆ Cliente envia mensagem ao servidor
 - ☆ Servidor envia resultado ao cliente
 - ☆ 2 send's
 - ☆ 2 receive's
 - ☆ 1 canal para cada cliente
 - ☆ Muitos clientes: muitos canais
 - RPC: apropriado para o modelo cliente-servidor

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto

28/45

Formas de Comunicação

Comunicação assíncrona

- Invocação por comando 'Call'
- Comunicação síncrona bidirecional

 - Cliente após invocar um 'call' é bloqueado até o retorno dos resultados

```

graph LR
    CC[Cliente Chamador] --> SC[Servidor Chamado]
    CC --> End(( ))
    style End fill:none,stroke:none
  
```

Processamento Paralelo e Distribuído - Prof.: Marcelo TrindadeRebonatto

29/45
