AULA Nº 06
Infraestrutura Comp. Alto Desempenho e Sist. Distribuídos

Middleware para Comp. Alto Desempenho

Julio Cezar Estrella
ICMC-USP

Introdução ao MPI

- **O que é Message Passing**
  - **O modelo de passagem de mensagens é um conjunto de processos que possuem acesso à memória local. As informações são enviadas da memória local do processo para a memória local do processo remoto**

Introdução ao MPI

- MPI
  - MPI é uma biblioteca de Message Passing desenvolvida para ambientes de memória distribuída, máquinas massivamente paralelas, NOWs (network of workstations) e redes heterogêneas.
  - Padrão criado por um consórcio entre empresas e universidades
  - É uma biblioteca de rotina que fornece funcionalidade básica para que os processos se comuniquem.
  - O paralelismo é explicito (programador é responsável pela distribuição)

Introdução ao MPI

- **Motivação**
  - Necessidade de pode computacional
  - Dificuldades na melhoria dos processadores sequenciais
  - Custo x Benefício
- **Dificuldades**
  - Comunicação, coordenação, sincronização
  - Distribuição de trabalho

Introdução ao MPI

- **Vantagens do MPI**
  - Maduro
  - Portabilidade é fácil
  - Combina eficientemente com diversos hardwares
    - Implementações proprietárias e públicas
  - Interface com usuário
    - Simples e eficiente
    - Buffer de manipulação
    - Permite abstrações de alto nível
  - Desempenho

Introdução ao MPI

- Desvantagens do MPI
  - Inclui muitas outras características além de passagem de mensagem
  - O ambiente de controle de execução depende da implementação

Introdução ao MPI

- **Características**
  - Thread safety
  - Comunicação ponto-a-ponto
    - Modos de comunicação: standard, synchronous, ready, buffered
    - Buffers estruturados
    - Tipos de dados derivados
  - Comunicação Coletiva
    - Nativo já incorporado e também operações coletivas definidas pelo usuário
    - Rotinas de movimentação de dados
  - Perfis
    - Usuários podem interceptar chamadas MPI e chamar suas próprias ferramentas

Introdução ao MPI

- **Modos de Comunicação**
  - **standard**: send não possui garantia de que a correspondente rotina receive foi iniciada
  - **synchronous**: send e receive podem iniciar antes de outra, mas precisam completar juntas
  - **ready**: usada para acesso de protocolos rápidos; garantia ao usuário que o correspondente receive foi postado.
  - **buffered:** copia os dados do buffer da mensagem para um buffer fornecido pelo usuário e retorna. Pode seguir com a computação e alterar o conteúdo original do buffer da mensagem , sabendo que essas alterações não vão refletir nos dados efetivamente enviados . Send pode iniciar e retornar antes de combinar com receive; espaço do buffer deve ser fornecidos

Introdução ao MPI

- Rotinas podem ser:
  - **Bloqueantes:** retornam quando estão localmente completas
    - O send não completa até que o buffer fique vazio
    - O receive não completa até que o buffer esteja cheio
    - Completar depende
      - Tamanho da mensagem
      - Quantidade de buffer no sistema

Introdução ao MPI

- **Não-Bloqueante**
  - Retorna imediatamente e permite e próximo passo a ser executado
  - Usado para sobrepor a comunicação e computação quando o tempo para enviar dados entre processo é grande
  - Retorna imediatamente “**request handle**”, que pode ser usado para uma busca ou aguardar
  - Término detectado por MPI_Wait ou MPI_Test()

Introdução ao MPI

- **Coletiva x Ponto-a-Ponto**
  - ***Ponto-a-ponto,bloqueante MPI_Send/MPI_Recv***
    - MPI_Send(start, count, datatype, dest, tag, comm)
    - MPI_Recv(start, count, datatype, source, tag, comm, status)
      - Simples, porém ineficiente
      - Maior parte do trabalho feito pelo processo 0
      - Coleta a saída dos processadores

Introdução ao MPI

- **Coletiva x Ponto-a-Ponto**
  - ***Operações coletivas para/de todos***
    - MPI_Bcast(start, count, datatype, root, comm)
    - MPI_Recv(start, result, count, datatype, source, operation, root, comm)
      - Chamada por todos os processos
      - Simples, compacta, mais eficiente
      - Deve ter o mesmo tamanho para count e datatype
      - O resultado tem significância somente sobre o nó 0

Introdução ao MPI

- **Quando usar MPI?**
  - Necessidade e programas paralelos portáveis
  - Escrever uma biblioteca paralela
- **Quando não usar MPI?**
  - Não necessita de paralelismo

Introdução ao MPI

- **Conceitos Básicos de MPI**
  - **Processo**
    - Cada parte do programa quebrado é chamado de processo. Os processos podem ser executados em uma única máquina ou em várias.
  - **Rank**
    - Todo o processo tem uma identificação única atribuída pelo sistema quando o processo é inicializado. Essa identificação é contínua representada por um número inteiro, começando de zero até N-1, onde N é o número de processos. Cada processo tem um rank, é ele é utilizado para enviar e receber mensagens.

Introdução ao MPI

- Grupos
  - Grupo é um conjunto ordenado de N processos. Todo e qualquergrupo é associado a um comunicador muitas vezes já predefinido como "MPI_COMM_WORLD“
  - O comunicador é um objeto local que representa o domínio (contexto) de uma comunicação (conjunto de processos que podem ser contatados).

Introdução ao MPI

- **Application Buffer**
  - É o endereço de memória, gerenciado pela aplicação, que armazena um dado que o processo necessita enviar ou receber
- **System Buffer**
  - É um endereço de memória reservado pelo sistema para armazenar mensagens.

Introdução ao MPI

- **Funções**
  - Mais de 125
- **Preciso usar todas?**
  - Para aprender MPI, NÃO
- Em geral são utilizadas:
  - **MPI_Init**
  - **MPI_Finalize**
  - **MPI_Comm_size**
  - **MPI_Comm_rank**
  - **MPI_Get_processor_name**
  - **MPI_Send / MPI_Bcast**
  - **MPI_Recv / MPI_Reduce**
- O restante é utilizada quando necessário → ***Flexibilidade***

Introdução ao MPI

- **MPI_Init**
  - Inicializa um processo MPI. Portanto, deve ser a primeira rotina a ser chamada por cada processo, pois estabelece o ambiente necessário para executar o MPI. Ela também sincroniza todos os processos na inicialização de uma aplicação MPI.

Introdução ao MPI

- **MPI_Finalize**
  - MPI_Finalize é chamada para encerrar o MPI. Ela deve ser a última função a ser chamada. É usada para liberar memória. Não existem argumentos.

Introdução ao MPI

- **Exemplo 1 (MPI_Init e MPI_Finalize)**

Código C:

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

  int main(int argc, char **argv)
{
  int  rc;
  rc = MPI_Init(&argc,&argv);
  if (rc == MPI_SUCCESS) {
    printf ("MPI iniciou
corretamente.\n");
    }

  MPI_Finalize();
  return 0;
  }
```

Saída com quatro processadores:

```
MPI iniciou corretamente.
MPI iniciou corretamente.
MPI iniciou corretamente.
MPI iniciou corretamente.
```

Introdução ao MPI

- **MPI_Comm_size**
  - Retorna o número de processos dentro de um grupo.
  - Obtém o comunicador como seu primeiro argumento e o endereço de uma variável inteira usada para retornar o número de processos.

Introdução ao MPI

- **MPI_Comm_rank**
  - Identifica um processo MPI dentro de um determinado grupo. Retorna sempre um valor inteiro entre 0 e n-1, onde n é o número de processos

Introdução ao MPI

- **Exemplo 2 (MPI_Comm_size e MPI_Comm_rank)**

Código C:

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

int main(int argc, char **argv)
 {
   int  numtasks, rank, rc;
   rc = MPI_Init(&argc,&argv);
   if (rc == MPI_SUCCESS){
      MPI_Comm_size (
                      MPI_COMM_WORLD,
                      &numtasks);
      MPI_Comm_rank(
                      MPI_COMM_WORLD,
                      &rank);
      printf ("Sou o processo %d de %d\n",
rank, numtasks);
     }
   MPI_Finalize(); return 0;
   }
```

Saída com quatro processadores:

```
Sou o processo 0 de 4
Sou o processo 3 de 4
Sou o processo 1 de 4
Sou o processo 2 de 4
```

Referências

Introduction to Parallel Computing, Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar - 2ª ed., Addison Wesley

Message Passing Interface (MPI)
https://computing.llnl.gov/tutorials/mpi/

Programação Paralela e Distribuída
http://www.dcc.fc.up.pt/~ricroc/aulas/1011/ppd/apontamentos/mpi.pdf

NetLab – Computação Paralela
http://netlab.ulusofona.pt/cp/

Próxima Aula

- **Middlewares para Computação Distribuída**