

MPI – Message Passing Interface

Uma breve introdução

Raphael Y. de Camargo
Ricardo Andrade

*Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo, Brasil*

São Paulo, 23 de novembro de 2007

O que é MPI?

- Especificação de API que facilita a criação de software paralelo
- Padrão criado por um consórcio envolvendo empresas e universidades
- Baseado no modelo de troca de mensagens
- Versões:
 - MPI-1 (1.2)
 - MPI-2 (2.1)

- Computação realizada por mais de um processador simultaneamente
- Motivação
 - Necessidade de poder computacional
 - Dificuldades na melhoria dos processadores sequenciais
 - Custo X Benefício
- Dificuldades
 - Comunicação, coordenação, sincronização
 - Distribuição de trabalho

- Modelo para computação paralela
 - Forma de estruturar um programa paralelo
- Conjunto de processos com memória local
- Comunicação por troca de dados (mensagens)
- Comunicação explícita através de operações cooperativas

Tipos de rotinas MPI-1

- Gerenciamento do ambiente MPI
- Comunicação ponto a ponto
- Comunicação coletiva
- Gerenciamento de grupos de processos
- Criação e manipulação de tipos de dados
- Criação de topologias virtuais
- Outras

Definições

- Rank: Número que identifica um processo em um grupo.
 - Número inteiro que inicia em 0
 - É utilizado para endereçar mensagens
- Grupo: Um conjunto de processos
 - Inicialmente todos processos pertencem a um grupo só
 - Associado a um comunicador
- Comunicador: Um identificador que define um grupo

- `MPI_Init(...)`
- `MPI_Finalize(...)`
- `MPI_Comm_rank(...)`
- `MPI_Comm_size(...)`
- `MPI_Send(...)`
- `MPI_Recv(...)`
- `MPI_Bcast(...)`
- `MPI_Reduce(...)`

MPI_Init & MPI_Finalize

- MPI_Init
 - Inicializa o sistema de troca de mensagens
 - Deve preceder todas as demais chamadas MPI
- MPI_Finalize
 - Finaliza o sistema de troca de mensagens
 - Deve ser a última chamada MPI
- Só podem ser chamadas um vez por processo

Exemplo 1

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.
```

- MPI_Comm_rank
 - Obtém o rank ou identificador de um processo dentro de um de um grupo
- MPI_Comm_size
 - Obtém a quantidade de processos pertencentes a um grupo

Exemplo 2

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
```

MPI_Send & MPI_Recv

- Comunicação ponto a ponto
- MPI_Send:
 - Envia uma mensagem a um destinatário
- MPI_Receive:
 - Recebe uma mensagem de um remetente

Mensagens

- São definidas por um conjunto de dados + envelope
- Dados: Escalares ou vetores + tipo
- Envelope: Destino, Origem, Tag, Comunicador

- Origem e destino:
 - Rank do processo que envia a mensagem e do destinatário
- Tag:
 - Identificador para a mensagem
 - As chamadas de MPI_Send e MPI_Recv correspondentes devem possuir mesmo tag
 - Identificador genérico: MPI_ANY_TAG
- Comunicador:
 - Identificador para o grupo, espaço de comunicação
 - As chamadas de MPI_Send e MPI_Recv correspondentes devem possuir mesmo comunicador
 - Identificador genérico: MPI_COMM_WORLD

Exemplo 3

Código C:

```
int main(int argc, char **argv) {
    int numtasks, rank, dest, source, rc, count, tag=1;
    char inmsg, outmsg; MPI_Status Stat;
    MPI_Init(&argc, &argv);

    ...

    if (rank == 0) {
        dest = 1; outmsg='x';
        printf("Enviando caractere %c para proc
%d\n", outmsg, dest);
        rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest,
tag, MPI_COMM_WORLD);
    }
    else if (rank == 1) {
        source = 0;
        rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source,
tag, MPI_COMM_WORLD, &Stat);
        printf("Recebi o caractere: %c do proc
%d\n", inmsg, source);
    }
    MPI_Finalize();
}
```

Saída com dois processadores:

Enviando caractere x para proc 1
Recebi o caractere x do proc 0

- MPI_Bcast
 - Envia uma mesma mensagem para todos os processos de um grupo
- MPI_Reduce
 - Aplica uma operação de redução em todos os processos de um grupo

Tipos de redução

- MPI_MAX: Máximo
- MPI_MIN: Mínimo
- MPI_SUM: Soma
- MPI_PROD: Produto
- MPI_LAND: "E" lógico
- MPI_BAND: "E" bit a bit
- MPI_LOR: "OU" lógico
- MPI_BOR: "OU" bit a bit
- MPI_LXOR: "OU" exclusivo
- MPI_BXOR: "OU" exclusivo bit a bit
- MPI_MAXLOC: Máximo + índice associado ao máximo
- MPI_MINLOC: Mínimo + índice associado ao mínimo

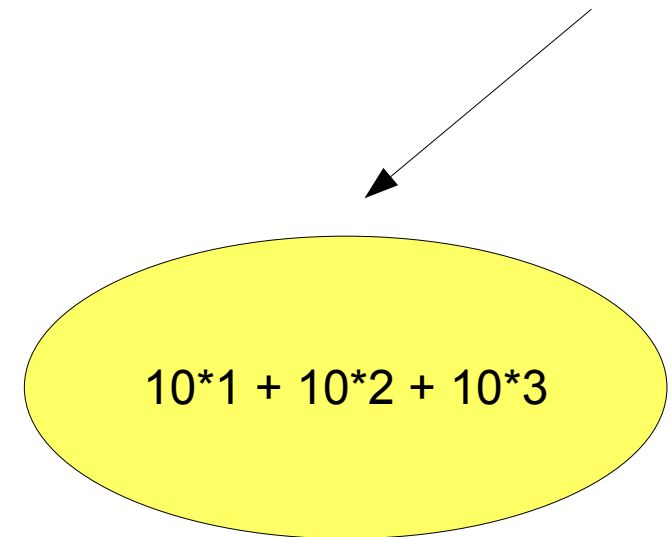
Exemplo 4

Código C:

```
int main(int argc, char **argv) {
    int rank, source=0, dest=0, rc;
    int value, totalValue;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,
    &rank);
    if (rank == 0) value = 10;
    MPI_Bcast (&value, 1, MPI_INT, source,
    MPI_COMM_WORLD);
    value *= rank + 1;
    MPI_Reduce(&value, &totalValue, 1,
    MPI_INT, MPI_SUM, dest,
    MPI_COMM_WORLD);
    if (rank == 0) printf ("Valor final calculado:
    %d\n", totalValue);
    MPI_Finalize();
    return 0;
}
```

Saída com três processadores:

Valor final calculado: 60



- PUMA ou *Pointwise Unconstrained Minimization Approach*
 - Método de otimização usado para recuperação de constantes óticas sistemas de filmes finos
- Parte do projeto TANGO
 - www.ime.usp.br/~egbirgin/tango/
- Estrutura *bag-of-tasks*
- Usa apenas 7 funções MPI diferentes

- Extensão do MPI
- Suporta comunicação por memória compartilhada (*One-Sided communications*)
- Suporta *multi-threading*
- Permite controle dinâmico de processos
- Define E/S Paralela

- Especificação MPI 1
 - <http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html>
- Especificação MPI 2
 - <http://www.mpi-forum.org/docs/mpi-20-html/mpi2-report.html>
- MPI no *Lawrence Livermore National Laboratory*
 - <http://www.llnl.gov/computing/tutorials/mpi/>
- MPI no *Argonne National Laboratory* (MPICH)
 - <http://www-unix.mcs.anl.gov/mpi>



InteGrade

- www.integrade.org.br
- Permite a execução de aplicações MPI através do MPICH (1.5)
- Suporte
 - integrade-support@incubadora.fapesp.br