Laboratório 13
Programação concorrente com memória distribuída usando MPI

Computação Concorrente (MAB-117)
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ — 28 de janeiro de 2016

Introdução

O objetivo deste Laboratório é introduzir os conceitos básicos de programação concorrente com memória distribuída usando o MPI[1]. Para cada atividade, siga o roteiro proposto e responda às questões colocadas. Este laboratório não valerá nota.

Atividade 1

Objetivo: Mostrar como iniciar e finalizar o ambiente para uso do MPI.

Roteiro:

1. Abra o arquivo **intro0.c** e entenda o que ele faz;
2. Compile o programa fazendo: **mpicc -o intro0 intro0.c**
3. Execute o programa com um único processo fazendo: **mpirun ./intro0**
4. Execute o programa com 2 processos fazendo: **mpirun -np 2 ./intro0**;
5. Execute o programa com 3 processos passando argumentos: **mpirun -np 3 ./intro0 ola 3 4**, o que mudou?
6. Observe e avalie os resultados.

Atividade 2

Objetivo: Identificar os processos de um grupo.

Roteiro:

1. Abra o arquivo **intro1.c** e entenda o que ele faz;
2. Compile o programa;
3. Execute o programa com um único processo;
4. Execute o programa **várias vezes** com 10 processos;
5. Observe e avalie os resultados.

Atividade 3

Objetivo: Enviar e receber mensagens.

Roteiro:

1. Abra o arquivo `intro2.c` e compreenda como ele funciona;
2. Execute o programa com um único processo;
3. Execute o programa várias vezes com 2 processos;
4. Execute o programa várias vezes com 4 processos;
5. Observe e avalie os resultados.

[1]*An Introduction to Parallel Programming*, Peter Pacheco, Morgan Kaufmann, 2011

Atividade 4

Objetivo: Enviar e receber mensagens (emissor desconhecido).

Roteiro:

1. Abra o arquivo **mpi_hello.c** e compreenda como ele funciona;
2. Execute o programa várias vezes com 2 processos;
3. Execute o programa várias vezes com 4 processos;
4. Observe e avalie os resultados.

Atividade 5

Objetivo: Calcular a integral de uma função usando a regra dos trapézios.

Roteiro:

1. Abra o arquivo **trap.c** e entenda como ele funciona;
2. Abra o arquivo **mpi_trap.c** e entenda como ele funciona;
3. **Complete os códigos das linhas 76 e 80**;
4. Compile e execute os dois programas com os mesmos parâmetros de entrada e compare os tempos de execução;
5. A solução com MPI obteve melhor desempenho?