Projeto 1
MPI_Reduce utilizando Sockets

1 Introdução

A função *MPI_Reduce* faz parte da biblioteca de comunicação paralela *MPI* (*Message Passing Interface*) e é usada em programas paralelos para realizar operações de redução em dados distribuídos entre múltiplos processos. Uma operação de redução refere-se à aplicação de uma operação (por exemplo, soma, produto, máximo, mínimo, etc.) sobre elementos de dados provenientes de diferentes processos *MPI* e, em seguida, acumular o resultado em um processo específico.

2 Descrição do projeto

Para a execução deste projeto, é necessário implementar a função de redução utilizando a biblioteca *sockets*. No entanto, diferentemente das implementações convencionais encontradas em bibliotecas como o *MPI*, onde cada nó trabalhador envia diretamente seu resultado ao nó gerenciador, será utilizado o conceito da técnica de barreira borboleta. Essa abordagem tem como objetivo reduzir o número de comunicações centralizadas no nó gerenciador.

Como observado na Figura 1, a cada etapa o número de comunicações é reduzida pela metade, até que no final um dos trabalhadores envia ao nó gerenciador o resultado final. O número de etapas, portanto, é proporcional ao logaritmo do número total de processos ($log2(N)$).

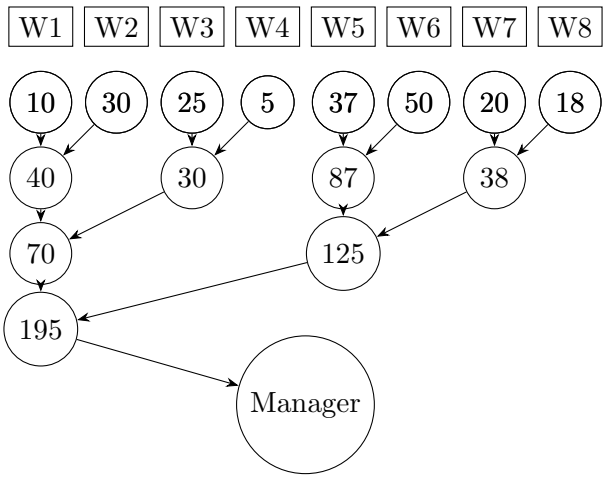

Figura 1: Exemplo da operação de redução utilizando o conceito da barreira borboleta, reduzindo a comunicação com o processo gerenciador.

Para a entrega do trabalho, deverá ser enviado o código-fonte de um programa que implemente a operação de redução de uma função à escolha do aluno (por exemplo, soma, produto, máximo, mínimo, etc.) aplicada a um conjunto de números gerados aleatoriamente.

- O sistema deverá ser composto por um **nó gerente** e **oito nós trabalhadores**.
- **Cada nó trabalhador deverá exibir o número gerado**, enquanto o **nó gerente será responsável por exibir o resultado final** da operação de redução.
- Cada processo deverá utilizar uma porta previamente conhecida por todos os nós, a fim de facilitar a troca de mensagens durante a execução da operação de redução.

  A aplicação deverá ser executada em uma única máquina, portanto, não é necessário especificar os endereços que cada serviço irá utilizar.

  Exemplo:

```
#define PORT_MANAGER 8080
#define PORT_WORKER0 8081
#define PORT_WORKER1 8082
#define PORT_WORKER2 8083
#define PORT_WORKER3 8084
#define PORT_WORKER4 8085
#define PORT_WORKER5 8086
#define PORT_WORKER6 8087
#define PORT_WORKER7 8088
```

  Com base nessas informações, cada nó deverá saber para quem enviar seus resultados a cada nível do processo. Por exemplo, conforme ilustrado na Figura 1:

  - No *primeiro nível*, o trabalhador 4 enviará seu resultado para o trabalhador 3, o qual deve esperar a mensagem do trabalhador 4.
  - No *segundo nível*, o trabalhador 3 enviará ao trabalhador 1 o resultado obtido após a operação de redução dos resultados dos trabalhadores 3 e 4.
  - Assim sucessivamente os outros trabalhadores deverão agir.