AULA 10

Programação de Sistemas Paralelos e Distribuídos

Envio Bloqueante: outras funções

- tamanho de mensagens: quando um programa recebe uma mensagem maior do que o especificado para o buffer de recepção, a mensagem é truncada ou uma condição de erro é gerada, ou ambos. É aceitável receber uma mensagem menor do que o especificado para o buffer de recepção. Neste caso, a aplicação pode requisitar o tamanho real da mensagem, usando MPI_Get_count.

  MPI_Get_count(MPI_Status *status,MPI_Data type dtype,int *count);

Envio Bloqueante: outras funções

- Probe: quando é impraticável pre-alocar um buffer de recepção, MPI_Probe sincroniza uma mensagem e retorna informações se realmente vai recebê-la. Não retorna até que uma mensagem seja sincronizada.

MPI_Probe(int source,int tag,MPI_Comm comm,MPI_Status *status);

Comunicação coletiva

- Operações coletivas consistem de várias mensagens ponto a ponto, envolvendo todos os processos de um dado comunicador.
- MPI_Bcast(void *buf,int count,MPI_Datatype dtype,int root,MPI_Comm comm);
- na operação broadcast todos os processos especificam o mesmo processo root (argumento root) cujo conteúdo do buffer será enviado. Os demais processos especificam buffers de recepção. Após a execução da chamada todos os buffers contêm os dados do buffer do processo root.

Comunicação coletiva

- MPI_Scatter(void *sendbuf,int sendcount,MPI_Datatype sendtype,void *recvbuf,int recvcount,MPI_Datatype recvtype,int root,MPI_Comm comm);
- Na operação scatter todos os processos especificam o mesmo count (número de elementos a serem recebidos). Argumentos send são significantes apenas para o processo root. O buffer de root contém, sendcount*N elementos do datatype tipo especificado. N é o número de processos no comunicador especificado.

Comunicação coletiva

- MPI_Gather(void *sendbuf,int sendcount,MPI_Datatype sendtype,void *recvbuf,int recvcount,MPI_Datatype recvtype,int root,MPI_Comm comm);
- a operação gather é o reverso da operação scatter (dados de buffers de todos processos para processo root)

Comunicação coletiva

- MPI_Reduce(void *sendbuf,void *recvbuf,int count,MPI_Datatype dtype,MPI_Op op,int root,MPI_Comm comm);
- Elementos dos buffers de envio são combinados par a par para um único elemento correspondente no buffer de recepção do root.
- Operaçoes de redução:
- MPI_MAX (maximum), MPI_MIN (minimum), MPI_SUM (sum), MPI_PROD (product), MPI_LAND (logical and), MPI_BAND (bitwise and) MPI_LOR (logical or) MPI_BOR (bitwise or) MPI_LXOR (logical exclusive or) MPI_BXOR (bitwise exclusive or)

Exercício 1

Busca da primeira ocorrencia de um número inteiro nos vetores A, B, C e D de inteiros com 100000 elementos.

Sugestão: 4 nós

No 0:

- envia o número a ser encontrado (utilizar MPI_Bcast) para todos nós
- recebe o índice da primeira ocorrencia , respectiva a cada matriz ,armazenando no vetor vet_ocorrencias (utilizar MPI_Gather)

Todos os nós:

- Inicializam a sua respectiva matriz (valores inteiros aleatórios)
- buscam a primeira ocorrencia do número especificado.
- Enviam o índice da primeira ocorrência (MPI_Gather)

Exercício 2

Encontrar o maior elemento (máximo) da matriz int A[1000][1000].

- Utilizar MPI_Scatter para distribuir as linha da matriz e MPI_Reduction para calcular o máximo total.

Funções de comunicação não bloqueantes

- chamadas não bloqueantes retornam imediatamente após o início da comunicação. O programador não sabe se o dado enviado já saiu do buffer de envio ou se o dado a ser recebido já chegou. Então, o programador deve verificar o seu estado antes de usar o buffer.

Comunicação não bloqueante

- envio: retorna após colocar o dado no buffer de envio.

MPI_Isend(void *buf,int count,MPI_Datatype dtype,int dest,int tag,MPI_Comm comm,MPI_Request *req)

req: objeto que contém informações sobre a mensagem, por exemplo o estado da mensagem.

Comunicação não bloqueante

- Recepção: é dado o início à operação de recepção e retorna. Não se deve ler os dados a serem recebidos, enquanto estes não estiverem disponíveis, o que pode ser verificado com a chamada da função MPI_Wait ou MPI_Test.

MPI_Irecv(void *buf,int count,MPI_Datatype dtype, int source, int tag, MPI_Comm comm, MPI_Request *req);

Comunicação não bloqueante

MPI_Wait(MPI_Request *req,MPI_Status *status);

Espera completar a transmissão ou a recepção.

MPI_Test(MPI_Request *req,int *flag,MPI_Status *status)

Retorna em flag a indicação se a transmissão ou recepção foi completada. Se true, o argumento status está preenchido com informação

MPI_Iprobe(int source,int tag,MPI_Comm comm,int *flag,MPI_Status *status);

Seta flag, indicando a presença do casamento da mensagem.

Exercício 3

Calcular:

W=(X+Y+Z) X T

X=A X B , Y=C X D, Z=E X F

W, T, X,Y,Z, A, B, C, D,E , F: matrizes double 500 X 500

Nó0 (tarefa1): inicializa A e B e calcula cada linha de X e envia para nó3 (tarefa4) (utilizar MPI_Isend e MPI_Wait)

Nó1´(tarefa2) e nó2 (tarefa3): inicializam C e D e E e F e ídem nó 0 para Y e Z, respectivamente.

Nó3 (tarefa4): inicializa T e recebe cada linha de X, Y e Z (MPI_Irecv e MPI_Wait), e calcula uma linha de W.