Programação para Sistemas Distribuídos e Paralelos

Laboratório 03 - MPI

Alunos:

- Gabriel Davi Silva Pereira 170010341
- João Pedro José Santos da Silva Guedes 170013910

Introdução

Para o laboratório 03, foi requerido a leitura de um arquivo de tamanho considerável, de forma que fosse contabilizado a quantidade de palavras com menos que 6 caracteres, quantidade de palavras que possuem entre 6 e 10 caracteres e, por fim, a quantidade de palavras acima de 10 caracteres.

Com isso, a solução para tal problema, foi proposto o uso de um sistema distribuído através do MPI. Dessa forma, há um sistema master responsável por ler o arquivo, e mandar a informação para os workers, onde servidores (workers), recebem a informação do arquivo e processam a quantidade de caracteres e palavras.

Descrição da Solução

Em uma solução convencional, o arquivo completo seria lido e executado no mesmo programa. Todavia, essa abordagem traria problemas de performance, uma vez que, para arquivos muito extensos, a leitura e contagem de palavras aconteceria em um único processo.

Logo, como solução para esse laboratório, o sistema foi dividido em 2 partes: Um serviço master e seus workers, onde o código verifica se o processo atual possui o id 0, e nesse caso atribui o papel de worker, caso contrário o processo é considerado como worker.

```
1  if(process_rank == MASTER_RANK){
2          execMaster();
3          return 0;
4      }
5
6      execWorker();
```

Figura 1 - Seleção entre master e worker

Dentro do Worker, o conteúdo do arquivo é lido e, então, os dados são divididos em partes para o envio para os workers. Onde inicialmente são enviados 128 bytes para o worker 1, 129 para o worker 2, e assim sucessivamente até chegar ao fim do arquivo ou chegar ao tamanho máximo de bytes (10.000.000), onde o código começa a enviar 128 bytes novamente.

```c
void execMaster(){
    clock_t begin = clock();

    // reading file
    FILE *file = fopen("file.txt", "r");
    if (file == NULL) exit(0);
    int chunkSize = CHUNK_SIZE_OFFSET;

    char *buffer = malloc(sizeof(char) * chunkSize);
    char c;
    int n = 0;

    int recieverRank = 1;

    while ((c = fgetc(file)) != EOF){
        if(n >= chunkSize){
            buffer[n] = '\0';
            sendMessageToWorker(buffer, chunkSize, recieverRank);
            updateRecieverRank(&recieverRank);
            updateChunkSize(&chunkSize);

            memset(buffer, 0, sizeof(*buffer)); // cleaning the data info
            buffer = realloc(buffer, chunkSize * sizeof(char)); // increasing the b
            n = 0;
            continue;
        }
        buffer[n++] = c;
    }
```

Figura 2 - Lendo dados do arquivo e enviado para o worker.

O worker recebe a mensagem enviada pelo processo master e então calcula o tamanho da mensagem utilizando o protocolo MPI_Probe em conjunto com o MPI_Get_count e, com esse tamanho calculado, ele aloca uma variável para receber a mensagem enviada pelo worker, calcula a quantidade de palavras e envia a resposta para o processo master.

```c
while(1){

        MPI_Probe(MASTER_RANK, TAG_2, MPI_COMM_WORLD, &status);
        MPI_Get_count(&status, MPI_CHAR, &msgSize);
        if(!msgSize) break;
        memset(msg, 0, sizeof(*msg));
        msg = realloc(msg, msgSize * sizeof(char));
        MPI_Recv(msg, msgSize, MPI_CHAR, MASTER_RANK, TAG_2, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        countWords(msg, msgSize,response);
        MPI_Send(response, 3, MPI_INT, MASTER_RANK, TAG_3, MPI_COMM_WORLD);
        memset(response, 0, sizeof(response));
    }

    int endResponse[3] = {0,0,0};
    MPI_Send(endResponse, 3, MPI_INT, MASTER_RANK, TAG_3, MPI_COMM_WORLD);
```

Figura 3 - Recebendo dados do processo master e calculando quantidade de palavras.

Para calcular o tempo gasto do programa a dupla optou por utilizar a função clock(), onde pegavam o tempo inicial da execução do programa menos o tempo final e tinham o resultado do tempo gasto.

```
sudjoao@MBP-de-Joao PSPD-Lab3 % mpirun -n 2 bin/bin.out
Less than 6: 165
More than 6 and Less than 10: 115
More than 10: 35
Exec time 0.00
```

Figura 4 - Exemplo de saída.

Problema de transferência de Buffer

O projeto foi executado em máquinas que possuem os processadores Apple M1 e Apple M2, e em ambas as máquinas a dupla não teve problemas com tamanho de buffer até 10.000.000 de caracteres, que era o proposto para a atividade.

```
sudjoao@MBP-de-Joao PSPD-Lab3 % mpirun -n 2 bin/bin.out
Less than 6: 11937523
More than 6 and Less than 10: 7540921
More than 10: 1966678
Exec time 3.24
```

Figura 5 - Lendo buffer com mais de 10.000.000 de caracteres.

Opinião Geral

Gabriel Davi:

- Opinião:

  É extremamente satisfatório trabalhar em soluções de problemas aplicáveis no mercado de trabalho. Esse laboratório foi muito legal para ter contato com uma tecnologia de programação paralela de mais "baixo nível" comparada com outras que trabalhamos na disciplina, pois consegui ver de forma clara como funcionava todo o processo.

- Contribuição:
  No geral, o trabalho foi feito em conjunto. Todavia, minha contribuição mais expressiva foi na confecção do laboratório, foi a criação das lógicas envolvendo o cálculo do tempo, palavras e também envio de mensagens.
- Nota: 10

João Pedro Guedes:

- Opinião

  Foi um projeto divertido de fazer, pois a lógica do MPI não é muito complicada, uma das partes mais interessantes foi descobrir algumas outras diretivas além da Recv e Send. Sinto que a utilização do MPI para compreender os conceitos básicos de sistemas distribuídos e paralelos é simplesmente sensacional.

- Contribuição:

  Contribui em toda a parte do processo, sendo a principal a pesquisa sobre uma diretiva para cálculo do tamanho da mensagem recebida. Além da lógica de envio de mensagens, leitura de arquivos etc.
- Nota: 10