

UNIVERSIDADE FEDERAL DE SANTA CATARINA – UFSC  
CENTRO TECNOLÓGICO - CTC  
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO  
PROGRAMAÇÃO PARALELA E DISTRIBUÍDA - INE5645  
LUÍS FERNANDO SILVEIRA - 19200644  
MAURICE ALEXSANDER ZANOTELLI - 19200647  
FLORIANÓPOLIS, SC - DEZEMBRO, 2022

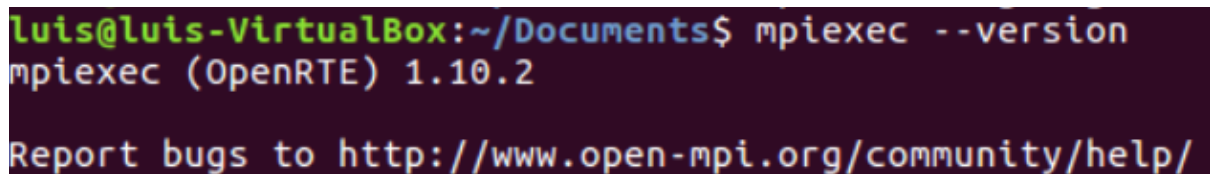
## TRABALHO FINAL - RELATÓRIO

### 1. INTRODUÇÃO

Este trabalho consiste em explorar estratégias de paralelização e programação distribuídas utilizando OpenMPI. O problema a ser resolvido é desenvolver um contador de ocorrências de cada palavra presente em uma lista, dado um arquivo de texto.

### 2. CONFIGURAÇÃO

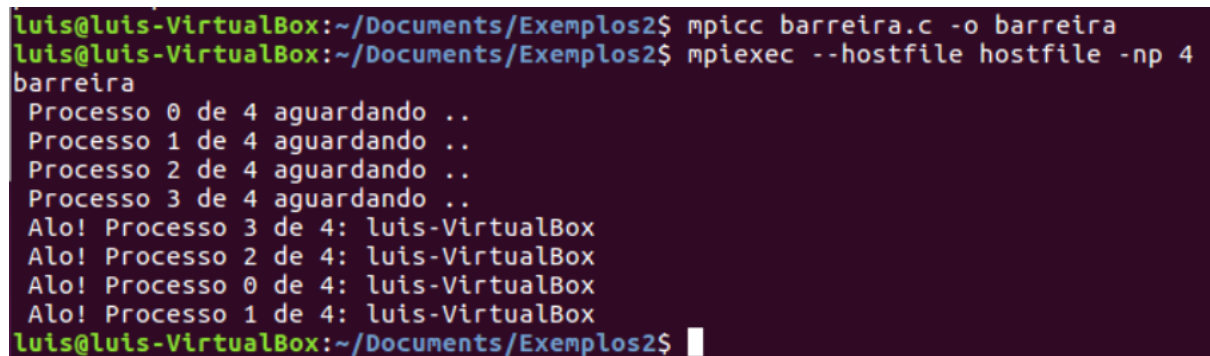
Para fazer uso das funções do OpenMPI é necessário antes fazer a instalação da implementação. Para tal, foi seguido os passos indicados nas vídeo aulas da matéria, assim como os materiais de apoio disponíveis.



```
luis@luis-VirtualBox:~/Documents$ mpirun --version
mpirun (OpenRTE) 1.10.2

Report bugs to http://www.open-mpi.org/community/help/
```

figura 1 - teste da instalação do OpenMPI



```
luis@luis-VirtualBox:~/Documents/Exemplos2$ mpicc barreira.c -o barreira
luis@luis-VirtualBox:~/Documents/Exemplos2$ mpirun --hostfile hostfile -np 4
barreira
Processo 0 de 4 aguardando ..
Processo 1 de 4 aguardando ..
Processo 2 de 4 aguardando ..
Processo 3 de 4 aguardando ..
Alo! Processo 3 de 4: luis-VirtualBox
Alo! Processo 2 de 4: luis-VirtualBox
Alo! Processo 0 de 4: luis-VirtualBox
Alo! Processo 1 de 4: luis-VirtualBox
luis@luis-VirtualBox:~/Documents/Exemplos2$
```

figura 2 - teste de execução do openMPI

### 3. ABORDAGEM

Anteriormente havia sido desenvolvido um programa com a mesma proposta de resolução, porém utilizando processamento sequencial. Neste primeiro trabalho, a estratégia utilizada para fazer o contador de palavras foi fazer um laço “for” que percorre todas as palavras da lista, e para cada palavra, o programa percorre o arquivo inteiro, contando o número de ocorrências da mesma.

```

/* chama a função para contar o numero de ocorrencias da palavra */
for (int i = 0; i < num_words; i++)
    qtd_word[i] = countOccurrences(keywords[i]);

for (int j = 0; j < num_words; j++)
    printf("Palavra: %s | Numero de vezes = %d.\n\n", keywords[j], qtd_word[j]);

```

figura 3 - função que percorre as palavras da lista e executa o contador de ocorrências.

```

int countOccurrences(char *word) {
    FILE *fptr = fopen("livro/mobdick.txt", "rb");

    /* Tratamento de excessao, caso nao cosiga ler o arquivo */
    if (fptr == NULL)
    {
        printf("Não foi possível abrir o arquivo.\n");
        exit(EXIT_FAILURE);
    }

    char str[BUFFER_SIZE];
    char *pos;

    int index;

    int count = 0;

    /* le as linhas do arquivo ate o final dele */
    while ((fgets(str, BUFFER_SIZE, fptr)) != NULL)
    {
        index = 0;

        /* encontra a proxima ocorrencia da palavra em str */
        while ((pos = strstr(str + index, word)) != NULL)
        {
            index = (pos - str) + 1;
            count++;
        }
    }

    return count;
}

```

figura 4 - função contador de ocorrências da palavra.

Partindo dessa ideia, foi definido como seriam implementadas as primitivas de comunicação ponto-a-ponto e primitivas de comunicação seletiva.

Para a primitiva ponto-a-ponto, o programa, após a inicialização do MPI, cria a lista de palavras a serem contadas (*keywords*), junto com uma outra lista de mesmo tamanho porém vazia (*recedor*), que receberá a lista de palavras quando ela for instanciada dentro do processo MPI\_Recv. O processo de *rank* 0 envia a lista de palavras para o processo de *rank* == 1 e o processo que recebe a lista executa a função de contar o número de ocorrências para cada palavra.

```

MPI_Comm_rank(MPI_COMM_WORLD, &world_rank); // Find out rank, size
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);

// We are assuming at least 2 processes for this task
if (world_size < 2) {
    fprintf(stderr, "World size must be greater than 1 for %s\n", argv[0]);
    MPI_Abort(MPI_COMM_WORLD, 1);
}

gettimeofday(&t1, NULL);

char keywords[WORD_COUNT][WORD_SIZE] = {
    "death", "day", "night", "fear", "sea", "whale", "ship", "joy",
    "king", "dispair", "tomorrow", "yesterday", "time", "home", "love", "snake",
    "man", "woman", "god", "tired", "me", "life", "rose", "book",
    "danger", "wind", "world", "flower", "star", "end", "python", "Roberto",
};
char recebedor[WORD_COUNT][WORD_SIZE];

if (world_rank == 0) {
    MPI_Send(&keywords, (WORD_COUNT * WORD_SIZE), MPI_BYTE, 1, 0, MPI_COMM_WORLD);
}
else {
    MPI_Recv(&recebedor, (WORD_COUNT * WORD_SIZE), MPI_BYTE, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    for (int i = 0; i < WORD_COUNT; i++)
        printf("Palavra: %s | Numero de vezes = %d.\n", recebedor[i], countOccurrences(recebedor[i]));
}

gettimeofday(&t2, NULL);
double t_total = (t2.tv_sec - t1.tv_sec) + ((t2.tv_usec - t1.tv_usec)/1000000.0);
printf("Tempo total de execucao = %f\n", t_total);

MPI_Finalize();

```

figura 5 - uso da primitiva ponto-a-ponto.

Para a primitiva de comunicação coletiva, o programa, após a inicialização do MPI, cria a lista de palavras a serem contadas(*keywords*), em seguida, a lista é dividida em partes iguais, de acordo com o número de palavras dentro da lista e a quantidade de processos que serão utilizados. A ideia é que cada processo pegue uma quantidade igual de palavras para contar. Então, o `MPI_Bcast` é invocado, e dentro dele, é definido o início e fim do pedaço da lista que o processo deverá contar. A partir disso, a função de contar as ocorrências de palavras é executada dentro desse intervalo.

```

MPI_Comm_rank(MPI_COMM_WORLD, &world_rank); // Find out rank, size
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
int begin, end;

gettimeofday(&t1, NULL);

// We are assuming at least 2 processes for this task
if (world_size < 2) {
    fprintf(stderr, "World size must be greater than 1 for %s\n", argv[0]);
    MPI_Abort(MPI_COMM_WORLD, 1);
}

char keywords[WORD_COUNT][WORD_SIZE] = {
    "death", "day", "night", "fear", "sea", "whale", "ship", "joy",
    "king", "dispair", "tomorrow", "yesterday", "time", "home", "love", "snake",
    "man", "woman", "god", "tired", "me", "life", "rose", "book",
    "danger", "wind", "world", "flower", "star", "end", "python", "Roberto",
};
int qtd_word = sizeof(keywords) / sizeof(keywords[0]);

MPI_Bcast(&keywords, (WORD_COUNT * WORD_SIZE), MPI_BYTE, 0, MPI_COMM_WORLD);

begin = qtd_word / world_size * world_rank;
end = qtd_word / world_size * (world_rank + 1);

for (int i = begin; i < end; i++)
    printf("Rank %d | Palavra: %s | Numero de vezes = %d\n", world_rank, keywords[i], countOccurrence(keywords[i]));

MPI_Barrier(MPI_COMM_WORLD);

if(world_rank == 0){
    gettimeofday(&t2, NULL);
    double t_total = (t2.tv_sec - t1.tv_sec) + ((t2.tv_usec - t1.tv_usec)/1000000.0);
    printf("Tempo total de execucao = %f\n", t_total);
}

MPI_Finalize();

```

figura 6 - uso da primitiva de comunicação coletiva.

#### 4. ANÁLISE DE DESEMPENHO

Foi proposto, para fim de análise de desempenho, que os programas fossem executados com tamanhos diferentes de lista de palavras. Primeiramente uma lista pequena (25 palavras), depois uma lista mediana (50 palavras) e por fim uma lista grande (100 palavras). Os resultados dessas execuções, junto com a análise do tempo decorrido em cada uma delas, para cada programa, são mostradas a seguir.

##### 4.1 PRIMITIVA PONTO-A-PONTO

O programa ponto-a-ponto se limita a utilizar apenas 2 processos (*rank* 0 e *rank* 1), por conta disso, é demonstrado abaixo uma execução para cada tamanho de lista, e os tempos de execução podem ser verificados ao final de cada execução.

```

luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpicc -o mpi mpi.c
luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 2 ./mpi
Palavra: death | Numero de vezes = 73.
Palavra: day | Numero de vezes = 303.
Palavra: night | Numero de vezes = 203.
Palavra: fear | Numero de vezes = 71.
Palavra: sea | Numero de vezes = 705.
Tempo total de execucao = 0.015705
Palavra: whale | Numero de vezes = 1334.
Palavra: ship | Numero de vezes = 697.
Palavra: joy | Numero de vezes = 38.
Palavra: king | Numero de vezes = 576.
Palavra: dispair | Numero de vezes = 0.
Palavra: tomorrow | Numero de vezes = 1.
Palavra: yesterday | Numero de vezes = 10.
Palavra: time | Numero de vezes = 578.
Palavra: home | Numero de vezes = 68.
Palavra: love | Numero de vezes = 55.
Palavra: snake | Numero de vezes = 3.
Palavra: man | Numero de vezes = 1302.
Palavra: woman | Numero de vezes = 10.
Palavra: god | Numero de vezes = 60.
Palavra: tired | Numero de vezes = 6.
Palavra: me | Numero de vezes = 5746.
Palavra: life | Numero de vezes = 176.
Palavra: rose | Numero de vezes = 39.
Palavra: book | Numero de vezes = 55.
Palavra: danger | Numero de vezes = 23.
Tempo total de execucao = 0.036076
luis@luis-VirtualBox:~/Documents/Trabalho Final$ █

```

figura 7 - tempo de execução fazendo uso da primitiva ponto-a-ponto em uma lista de 25 palavras.

```

Palavra: life | Numero de vezes = 176.
Palavra: rose | Numero de vezes = 39.
Palavra: book | Numero de vezes = 55.
Palavra: danger | Numero de vezes = 23.
Tempo total de execucao = 0.072026
luis@luis-VirtualBox:~/Documents/Trabalho Final$ █

```

figura 8 - tempo de execução fazendo uso da primitiva ponto-a-ponto em uma lista de 50 palavras.

```

Palavra: life | Numero de vezes = 176.
Palavra: rose | Numero de vezes = 39.
Palavra: book | Numero de vezes = 55.
Palavra: danger | Numero de vezes = 23.
Tempo total de execucao = 0.241371
luis@luis-VirtualBox:~/Documents/Trabalho Final$ █

```

figura 9 - tempo de execução fazendo uso da primitiva ponto-a-ponto em uma lista de 100 palavras.

Número de processos	Quantidade de palavras	Tempo de execução (s)
2	25	0,036076
2	50	0,072026
2	100	0,241371

Podemos verificar que o resultado está dentro do esperado, o menor tempo de execução ocorreu com 25 palavras, e o aumento no tempo para 50 palavras foi linear. Já para o programa com 100 palavras, notamos que o aumento no tempo de execução foi mais acentuado.

## 4.2 PRIMITIVA COMUNICAÇÃO COLETIVA

Para o programa com comunicação coletiva foram feitos três testes para cada tamanho da lista de palavras. Primeiramente executando com dois processos configurados, em seguida com quatro e por fim com oito processos. Os tempos de execução estão demonstrados abaixo.

Lista com 25 palavras:

```
luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 2 bcast
Rank 1 | Palavra: time | Numero de vezes = 578
Rank 0 | Palavra: death | Numero de vezes = 73
Rank 1 | Palavra: home | Numero de vezes = 68
Rank 1 | Palavra: love | Numero de vezes = 55
Rank 0 | Palavra: day | Numero de vezes = 303
Rank 1 | Palavra: snake | Numero de vezes = 3
Rank 0 | Palavra: night | Numero de vezes = 203
Rank 1 | Palavra: man | Numero de vezes = 1302
Rank 0 | Palavra: fear | Numero de vezes = 71
Rank 1 | Palavra: woman | Numero de vezes = 10
Rank 0 | Palavra: sea | Numero de vezes = 705
Rank 1 | Palavra: god | Numero de vezes = 60
Rank 0 | Palavra: whale | Numero de vezes = 1334
Rank 1 | Palavra: tired | Numero de vezes = 6
Rank 0 | Palavra: ship | Numero de vezes = 697
Rank 1 | Palavra: me | Numero de vezes = 5746
Rank 1 | Palavra: life | Numero de vezes = 176
Rank 0 | Palavra: joy | Numero de vezes = 38
Rank 0 | Palavra: king | Numero de vezes = 576
Rank 1 | Palavra: rose | Numero de vezes = 39
Rank 0 | Palavra: dispair | Numero de vezes = 0
Rank 1 | Palavra: book | Numero de vezes = 55
Rank 0 | Palavra: tomorrow | Numero de vezes = 1
Rank 0 | Palavra: yesterday | Numero de vezes = 10
Tempo total de execucao = 0.073370
luis@luis-VirtualBox:~/Documents/Trabalho Final$
```

figura 10 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 25 palavras com 2 processos .

```
luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpicc bcast.c -o bcast
luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 4 bcast
Rank 2 | Palavra: time | Numero de vezes = 578
Rank 2 | Palavra: home | Numero de vezes = 68
Rank 2 | Palavra: love | Numero de vezes = 55
Rank 3 | Palavra: god | Numero de vezes = 60
Rank 2 | Palavra: snake | Numero de vezes = 3
Rank 3 | Palavra: tired | Numero de vezes = 6
Rank 2 | Palavra: man | Numero de vezes = 1302
Rank 3 | Palavra: me | Numero de vezes = 5746
Rank 2 | Palavra: woman | Numero de vezes = 10
Rank 3 | Palavra: life | Numero de vezes = 176
Rank 3 | Palavra: rose | Numero de vezes = 39
Rank 3 | Palavra: book | Numero de vezes = 55
Rank 1 | Palavra: ship | Numero de vezes = 697
Rank 0 | Palavra: death | Numero de vezes = 73
Rank 1 | Palavra: joy | Numero de vezes = 38
Rank 0 | Palavra: day | Numero de vezes = 303
Rank 0 | Palavra: night | Numero de vezes = 203
Rank 1 | Palavra: king | Numero de vezes = 576
Rank 1 | Palavra: dispair | Numero de vezes = 0
Rank 0 | Palavra: fear | Numero de vezes = 71
Rank 0 | Palavra: sea | Numero de vezes = 705
Rank 1 | Palavra: tomorrow | Numero de vezes = 1
Rank 0 | Palavra: whale | Numero de vezes = 1334
Rank 1 | Palavra: yesterday | Numero de vezes = 10
Tempo total de execucao = 0.107773
luis@luis-VirtualBox:~/Documents/Trabalho Final$
```



figura 11 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 25 palavras com 4 processos.

```
luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 8 bcast
Rank 5 | Palavra: snake | Numero de vezes = 3
Rank 5 | Palavra: man | Numero de vezes = 1302
Rank 6 | Palavra: god | Numero de vezes = 60
Rank 4 | Palavra: time | Numero de vezes = 578
Rank 0 | Palavra: death | Numero de vezes = 73
Rank 6 | Palavra: tired | Numero de vezes = 6
Rank 2 | Palavra: ship | Numero de vezes = 697
Rank 5 | Palavra: woman | Numero de vezes = 10
Rank 4 | Palavra: home | Numero de vezes = 68
Rank 2 | Palavra: joy | Numero de vezes = 38
Rank 0 | Palavra: day | Numero de vezes = 303
Rank 1 | Palavra: fear | Numero de vezes = 71
Rank 6 | Palavra: me | Numero de vezes = 5746
Rank 0 | Palavra: night | Numero de vezes = 203
Rank 2 | Palavra: king | Numero de vezes = 576
Rank 4 | Palavra: love | Numero de vezes = 55
Rank 1 | Palavra: sea | Numero de vezes = 705
Rank 3 | Palavra: dispair | Numero de vezes = 0
Rank 7 | Palavra: life | Numero de vezes = 176
Rank 3 | Palavra: tomorrow | Numero de vezes = 1
Rank 1 | Palavra: whale | Numero de vezes = 1334
Rank 7 | Palavra: rose | Numero de vezes = 39
Rank 3 | Palavra: yesterday | Numero de vezes = 10
Rank 7 | Palavra: book | Numero de vezes = 55
Tempo total de execucao = 0.145443
luis@luis-VirtualBox:~/Documents/Trabalho Final$
```

figura 12 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 25 palavras com 8 processos.

Lista com 50 palavras:

```
luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 2 bcast
Rank 1 | Palavra: death | Numero de vezes = 73
Rank 0 | Palavra: life | Numero de vezes = 176
Rank 1 | Palavra: life | Numero de vezes = 176
Rank 1 | Palavra: rose | Numero de vezes = 39
Rank 0 | Palavra: rose | Numero de vezes = 39
Rank 0 | Palavra: book | Numero de vezes = 55
Rank 1 | Palavra: book | Numero de vezes = 55
Rank 1 | Palavra: danger | Numero de vezes = 23
Rank 0 | Palavra: danger | Numero de vezes = 23
Tempo total de execucao = 0.127730
luis@luis-VirtualBox:~/Documents/Trabalho Final$
```

figura 13 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 50 palavras com 2 processos.

```
luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 4 bcast
Rank 2 | Palavra: danger | Numero de vezes = 23
Rank 3 | Palavra: me | Numero de vezes = 5746
Rank 3 | Palavra: life | Numero de vezes = 176
Rank 1 | Palavra: me | Numero de vezes = 5746
Rank 1 | Palavra: life | Numero de vezes = 176
Rank 0 | Palavra: yesterday | Numero de vezes = 10
Rank 3 | Palavra: rose | Numero de vezes = 39
Rank 1 | Palavra: rose | Numero de vezes = 39
Rank 1 | Palavra: book | Numero de vezes = 55
Tempo total de execucao = 0.156048
luis@luis-VirtualBox:~/Documents/Trabalho Final$
```

figura 14 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 50 palavras com 4 processos.

```

luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 8 bcast
Rank 6 | Palavra: yesterday | Numero de vezes = 10

Rank 3 | Palavra: life | Numero de vezes = 176
Rank 1 | Palavra: yesterday | Numero de vezes = 10
Rank 0 | Palavra: sea | Numero de vezes = 705
Rank 7 | Palavra: life | Numero de vezes = 176
Rank 3 | Palavra: rose | Numero de vezes = 39
Rank 0 | Palavra: whale | Numero de vezes = 1334
Rank 7 | Palavra: rose | Numero de vezes = 39
Rank 3 | Palavra: book | Numero de vezes = 55
Tempo total de execucao = 0.276044
luis@luis-VirtualBox:~/Documents/Trabalho Final$

```

figura 15 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 50 palavras com 8 processos.

Lista com 100 palavras:

```

luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 2 bcast
Rank 1 | Palavra: death | Numero de vezes = 73

Rank 1 | Palavra: life | Numero de vezes = 176
Rank 1 | Palavra: rose | Numero de vezes = 39
Rank 0 | Palavra: life | Numero de vezes = 176
Rank 0 | Palavra: rose | Numero de vezes = 39
Rank 1 | Palavra: book | Numero de vezes = 55
Rank 0 | Palavra: book | Numero de vezes = 55
Rank 0 | Palavra: danger | Numero de vezes = 23
Rank 1 | Palavra: danger | Numero de vezes = 23
Tempo total de execucao = 0.260720
luis@luis-VirtualBox:~/Documents/Trabalho Final$

```

figura 16 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 100 palavras com 2 processos.

```

luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 4 bcast
Rank 2 | Palavra: death | Numero de vezes = 73

Rank 0 | Palavra: life | Numero de vezes = 176
Rank 1 | Palavra: life | Numero de vezes = 176
Rank 1 | Palavra: rose | Numero de vezes = 39
Rank 0 | Palavra: rose | Numero de vezes = 39
Rank 1 | Palavra: book | Numero de vezes = 55
Rank 1 | Palavra: danger | Numero de vezes = 23
Rank 0 | Palavra: book | Numero de vezes = 55
Rank 0 | Palavra: danger | Numero de vezes = 23
Tempo total de execucao = 0.324231
luis@luis-VirtualBox:~/Documents/Trabalho Final$

```

figura 17 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 100 palavras com 4 processos.



```

luis@luis-VirtualBox:~/Documents/Trabalho Final$ mpiexec --hostfile hostfile -np 8 bcast
Rank 5 | Palavra: tomorrow | Numero de vezes = 1
Rank 2 | Palavra: king | Numero de vezes = 576
Rank 2 | Palavra: dispair | Numero de vezes = 0
Rank 0 | Palavra: tomorrow | Numero de vezes = 1
Rank 3 | Palavra: life | Numero de vezes = 176
Rank 4 | Palavra: dispair | Numero de vezes = 0
Rank 2 | Palavra: tomorrow | Numero de vezes = 1
Rank 3 | Palavra: rose | Numero de vezes = 39
Rank 0 | Palavra: yesterday | Numero de vezes = 10
Tempo total de execucao = 0.391272
luis@luis-VirtualBox:~/Documents/Trabalho Final$ █

```

figura 18 - tempo de execução fazendo uso da primitiva de comunicação coletiva em uma lista de 100 palavras com 8 processos.

Número de processos	Quantidade de palavras	Tempo de execução (s)
2	25	0,073370
4	25	0,107773
8	25	0,145443
2	50	0,127730
4	50	0,156048
8	50	0,276044
2	100	0,260720
4	100	0,324231
8	100	0,391272

Nestes testes percebe-se que para o escopo desse problema, os melhores tempos aparecem quando utilizamos apenas dois processos trabalhando coletivamente. Para as execuções com as listas de cinquenta e cem palavras, há um aumento no tempo de execução, porém esse aumento é consideravelmente menor em relação ao que ocorreu no programa ponto-a-ponto.

## 5. CONCLUSÃO

O desenvolvimento deste trabalho serviu ao propósito de compreender a aplicação da linguagem MPI para programação paralela e distribuída. Dentro do escopo deste trabalho percebemos que o uso dessa técnica tem ganho de performance em condições específicas, mas pode gerar perda de rendimento em outras. Fazendo a análise de desempenho é natural concluir que os resultados apresentados estão diretamente relacionados ao nível de complexibilidade do problema, e que poderíamos ter conclusões diferentes caso o tamanho do arquivo a ser lido fosse muito maior, por exemplo.