

UNIVERSIDADE LUSÓFONA DO PORTO

TRABALHO FINAL 6

PROGRAMAÇÃO PARALELA
MPI

NUNO NOGUEIRA PONTES - A21200373

RAFAEL MAGALHÃES - A21502512

ÍNDICE

1.	INTRODUÇÃO.....	5
2.	ENUNCIADO	5
3.	O QUE É O MPI.....	5
4.	SISTEMAS PARALELOS.....	6
5.	EVOLUÇÃO DO MPI.....	7
6.	DIFERENTES IMPLEMENTAÇÕES EM MPI.....	8
7.	SINTAXE DE FUNÇÕES EM MPI	8
8.	COMMUNICATOR MPI.....	9
9.	FUNÇÕES MPI	10
10.	COMPILAR E EXECUTAR PROGRAMAS EM MPI.....	11
11.	MPI DATATYPES	12
12.	IMPLEMENTAÇÃO DE TRÊS ALGORÍTMOS COMO EXEMPLO	13
13.	IMPLEMENTAR A BIBLIOTECA DO MPI NO WINDOWS.....	13
14.	CONCLUSÕES E OBSERVAÇÕES	17
15.	REFERENCIAS WEB GRÁFICAS	18

ÍNDICE DE TABELAS

TABELA 1 – MPI-1.X	7
TABELA 2 – MPI-2.X	7
TABELA 3 – MPI-3.0.....	7
TABELA 4 - MÓDULOS.....	11
TABELA 5 – COMPILAÇÃO EM C	12
TABELA 6 – EXECUTAR O PROGRAMA	12
TABELA 7 – TIPOS DE DADOS MPI	13

ÍNDICE DE FIGURAS

FIGURA 1 – SISTEMAS PARALELOS -----	6
FIGURA 2 - MPI_ -----	8
FIGURA 3 – MPI_INIT-----	8
FIGURA 4 – MPI_FINALIZE -----	9
FIGURA 5 – MPI_COMM_RANK() -----	9
FIGURA 6 – CONSTANTES PREDEFINIDAS -----	9
FIGURA 7 – INT MPI_INIT() -----	10
FIGURA 8 – INT MPI_FINALIZED() -----	10
FIGURA 9 – INT MPI_COMM_SIZE() -----	10
FIGURA 10 – INT MPI_COMM_RANK()-----	10
FIGURA 11 – INT MPI_ABORT() -----	11
FIGURA 12 – DIRETÓRIO ADICIONAL MPI INCLUDE -----	13
FIGURA 13 – DIRETÓRIO ADICIONAL MPI LIB -----	14
FIGURA 14 – DIRETÓRIO ADICIONAL MPI BIN NO PATH DO WINDOWS -----	15
FIGURA 15 - MPIEXEC -----	15
FIGURA 16 – VALIDAR REGISTO -----	15
FIGURA 17 – PROGRAMA SOMA 6 PRIMEIROS NÚMEROS -----	16
FIGURA 18 – SOMA 100 NÚMEROS -----	16
FIGURA 19 – ERRO GATHER() -----	16
FIGURA 20 – MULTIPLICAÇÃO DE DUAS MATRIZES -----	17

1. INTRODUÇÃO

Este trabalho 6, foi realizado no âmbito da disciplina de programação paralela, onde temos como objetivo primordial elaborar uma pesquisa exhaustiva sobre a biblioteca de computação paralela do MPI, e posteriormente a isso apresentar problemas matemáticos em que a biblioteca nos possa auxiliar.

Assim como toda a implementação necessário para executar programas do MPI pela linha de comandos.

2. ENUNCIADO

Enunciado para trabalho 6
Estudo sobre o funcionamento de aplicação das bibliotecas MPI.
Exemplo de utilização das bibliotecas aplicas à resolução de um problema.
Apresentação na aula – grupos de 2 a 3 alunos.

Tabela 1 – Enunciado para o trabalho 6

3. O QUE É O MPI

MPI (Message Passing Interface), consiste no padrão para implementar a comunicação entre nós de computação/processos em programação paralela. O mesmo define a sintaxe e a semântica de um nucelo de rotinas de biblioteca.

O MPI não é uma linguagem de programação, consiste apenas numa biblioteca de definições e funções que podem ser implementadas em linguagens de programação tais como C, Fortran, Python.

Um dos grandes objetivos do MPI é ser eficiente, flexível, pratico e portátil.

4. SISTEMAS PARALELOS

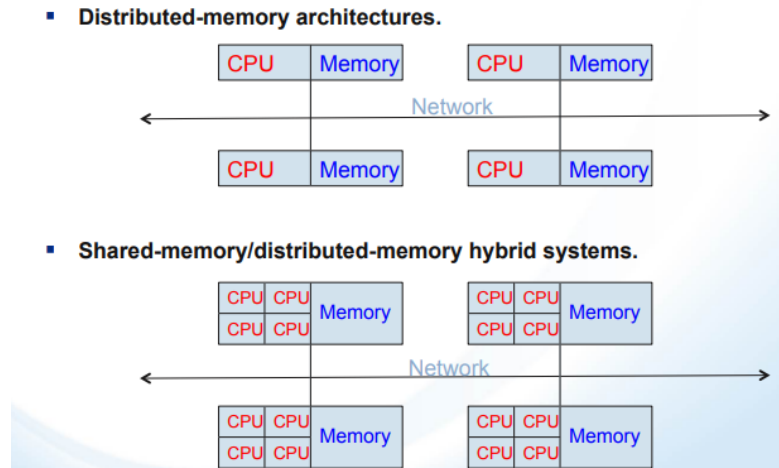


Figura 1 – Sistemas Paralelos

Em sistemas de memória compartilhada, o MPI fornece o entronó de rotinas de passagem de mensagens, como por exemplo, um processo MPI por nó.

Em sistemas híbridos, existem duas formas de se configurar o MPI. Usar o MPI para comunicação entre nós e o OpenMP para o compartilhamento das tasks entre nós. Neste caso, executa-se um processo MPI por nó.

A maioria dos sistemas HPC atuais são sistemas híbridos.

5. EVOLUÇÃO DO MPI

MPI-1.x:
O objetivo era desenvolver um padrão amplamente utilizado para escrever software de “messagepassing”.
Inicialmente, mais de 40 organizações participaram da discussão.
O relatório final da versão 1.0 foi concluído em 1994.

Tabela 2 – MPI-1.x

MPI-2.x:
Contém correções e extensões para MPI-1.x.
Focado na criação e gestão de processos, comunicações unilaterais, comunicações coletivas, interfaces externas e input/output paralelo.
O relatório final da versão 1.0 foi concluído em 1994.

Tabela 3 – MPI-2.x

MPI-3.0:
Extensões principais ao MPI, incluindo nonblocking collectives, novas operações de comunicações one-sided unilateral e ligações Fortran 2008.
Publicado em 2012.

Tabela 4 – MPI-3.0

6. DIFERENTES IMPLEMENTAÇÕES EM MPI

Para a pesquisa, neste trabalho 6, em específico, utilizou-se o auxílio do MPICH (em concreto MPICH2), que consiste em uma implementação open-source, ao qual se pode fazer o download, livres de quaisquer custos adicionais. O local utilizado, encontra-se nas referências webgráficas deste trabalho.

Do website onde se retirou a biblioteca do MPICH, existem outras variantes, tais como MPICH2, que consiste na versão atualizada do MPI.

Existem outras alternativas, no entanto não são open-source, logo encarecem do ponto de vista monetário. Tais como Intel MPI ou HP MPI.

7. SINTAXE DE FUNÇÕES EM MPI

As funções MPI, data types e constantes predefinidas começam todas com MPI_

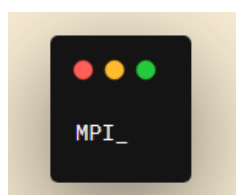


Figura 2 - MPI_

As funções MPI começam com uma letra maiúscula, seguida por letras minúsculas. Como por exemplo:

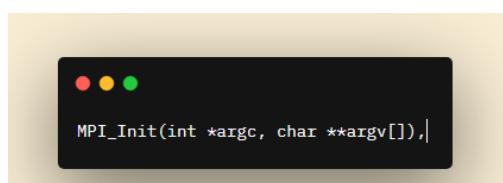
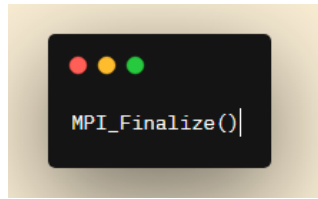


Figura 3 – MPI_Init



**Figura 4 –
MPI_Finalize**

Geralmente usa-se o `MPI_finalize()` no final do algoritmo, normalmente antecedido de um `return 0`, caso a linguagem de programação seja C. O seu objetivo, como o nome sugere, termina a execução do ambiente MPI.

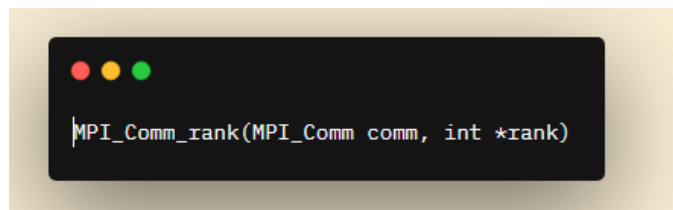
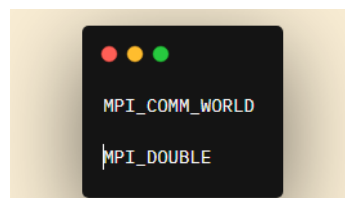


Figura 5 – MPI_Comm_rank()

Determina o rank do processo chamado, onde `size-1`, em que o `size` representa o valor de `MPI_COMM_SIZE`.

O nome de constantes MPI predefinidas são todas maiúsculas. Como por exemplo:



**Figura 6 – Constantes
Predefinidas**

8. COMMUNICATOR MPI

Um comunicador (communicator), conjunto de processos que podem enviar mensagens entre si.

`MPI_COMM_WORLD` é um comunicador predefinido que consiste em todos os processos em execução quando o programa é executado.

9. FUNÇÕES MPI

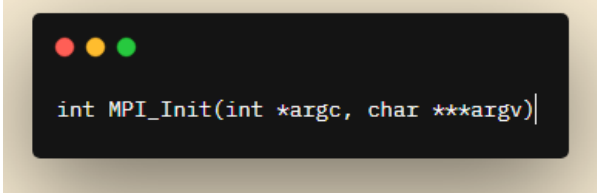
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The text `int MPI_Init(int *argc, char ***argv)|` is displayed in a monospaced font.

Figura 7 – Int MPI_Init()

Inicializa o ambiente de execução MPI. Esta função requer se chamada em cada programa MPI, deve ser chamada antes de qualquer outra função MPI e deve ser chamada apenas uma vez em um programa MPI.

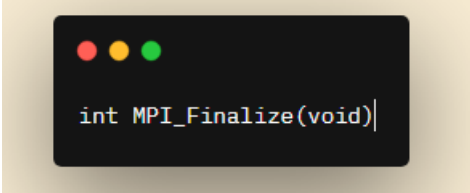
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The text `int MPI_Finalize(void)|` is displayed in a monospaced font.

Figura 8 – Int MPI_Finalized()

Encerra o ambiente de execução MPI. Efetivamente a última rotina MPI chamada em um programa MPI. Nenhuma outra rotina MPI pode ser chamada depois disso.

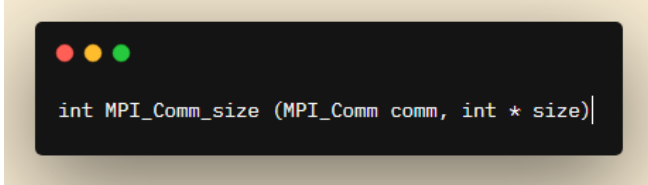
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The text `int MPI_Comm_size (MPI_Comm comm, int * size)|` is displayed in a monospaced font.

Figura 9 – int MPI_Comm_size()

Determina o número total de processos no comunicador especificado, como MPI_COMM_WORLD.

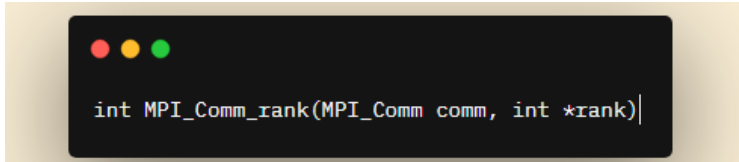
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The text `int MPI_Comm_rank(MPI_Comm comm, int *rank)|` is displayed in a monospaced font.

Figura 10 – int MPI_Comm_rank()

Retorna o rank do processo MPI de chamada com o comunicador especificado. Cada processo MPI é atribuído a um valor inteiro único entre 0 e (número de processos - 1)

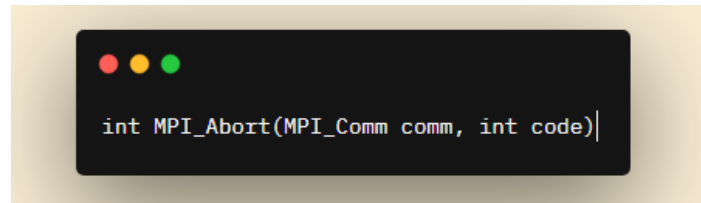


Figura 11 – `int MPI_Abort()`

Encerra os processos MPI associados à comunicação do comunicador e retorna o código de erro especificado no código.

10. COMPILAR E EXECUTAR PROGRAMAS EM MPI

O comando `mpicc` é um script da shell que define o ambiente para compilar um programa MPI usando o compilador GNU-C `mpirun` ou `mpiexec` executa o programa MPI.

Para este relatório, em específico o comando que mais se utilizou foi o `mpiexec`, devido a versão do MPI que utilizamos, nomeadamente MPICH2. Em parte, também porque a experiência foi efetuada em ambiente Windows, e qualquer outro comando a não ser o `mpiexec` originava erros, logo optou-se por utilizar apenas este. Num futuro próximo, utilizar-se-á um sistema operativo mais conveniente para o desenvolvimento de futuros testes, a nível de paralelização.

No `hpc`, as variáveis de ambiente são definidas usando os módulos:

Módulos
module avail (módulos existentes no sistema).
module list (módulos atualmente carregados para o utilizador)
module load <nome do módulo> (carrega um novo módulo)

Tabela 5 - Módulos

Para compilar-se um programa MPI na linguagem de programação C, utiliza-se o seguinte comando.

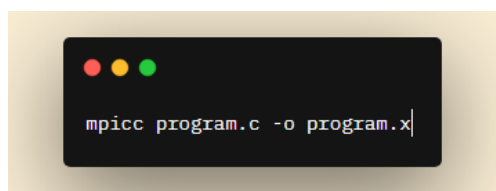


Tabela 6 – Compilação em C

Refere-se que, no nosso contexto de trabalho, não foi utilizado este tipo de compilação, isto porque utilizamos IDE para compilar os programas em MPI. Como referido, o nosso ambiente de trabalho foi Windows, pelo que ficamos restringidos a algumas potencialidades, assim como alguns erros, que de outra forma em Linux, não existiriam.

Para executar um programa em MPI, utilizou-se o seguinte comando.

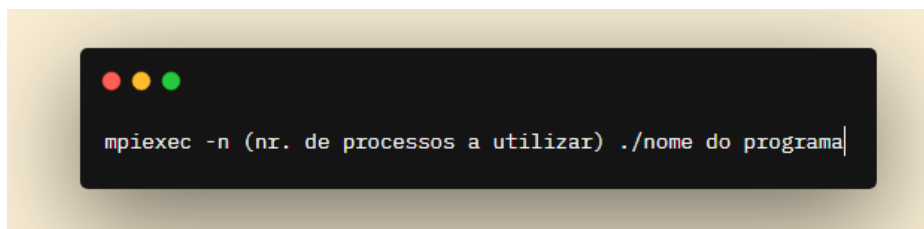


Tabela 7 – Executar o programa

Como referido este comando funciona no Windows.

Diferentes sistemas computacionais dados que são configurados de maneira diferente. Tivemos que efetivamente de nos certificar que a documentação do sistema ou guia do utilizador para comandos e opções MPI dispunha de soluções para o nosso OS.

11. MPI DATATYPES

MPI Datatype	C Datatype
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double

MPI_DOUBLE	long double
MPI_LONG_DOUBLE	
MPI_BYTE	
MPI_PACKED	

Tabela 8 – Tipos de dados MPI

11.1 IMPLEMENTAÇÃO DE TRÊS ALGORITMOS COMO EXEMPLO

Para este capítulo efetuaram-se testes a nível de implementação de código a três algoritmos diferentes. Nomeadamente, a uma multiplicação entre duas matrizes, soma dos primeiros 6 números em sequência e somar 100 números.

12. IMPLEMENTAR A BIBLIOTECA DO MPI NO WINDOWS

Para se instalar a biblioteca do MPICH2, optou-se em primeiro lugar recorrer ao site da Microsoft. Onde foram disponibilizados dois ficheiros executáveis, nomeadamente um ficheiro que instala por defeito no diretório dos programas (x86) do computador, na pasta Microsoft SDKs. Foi feita essa instalação, e posteriormente adicionadas as dependências, nomeadamente a pasta do include, como se pode confirmar na figura abaixo.

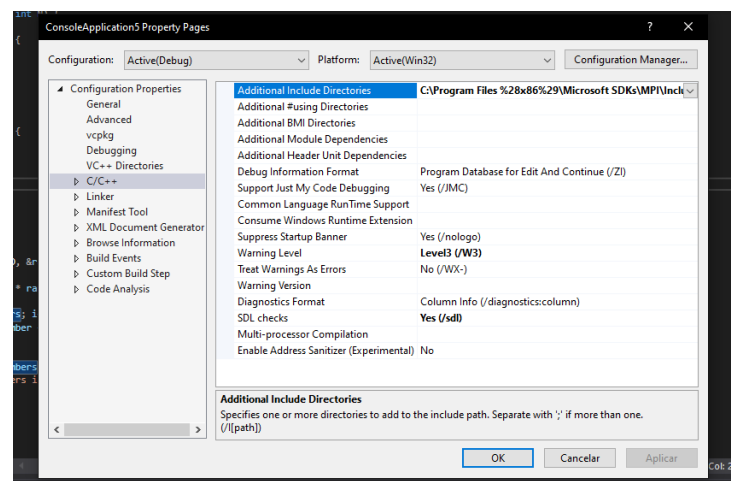


Figura 12 – Diretório adicional MPI Include

No entanto mesmo após a instalação dos dois executáveis do MPI, e posteriormente a inclusão dos diretórios dentro do Visual Studio, os erros subsistiam.

Confirmou-se com toda a certeza que a versão instalada era definitivamente a correta para o Windows.

Tentou-se contornar a situação adicionando mais uma dependência ao Visual Studio, nomeadamente ao Linker, onde se adicionou o diretório da lib do MPI.

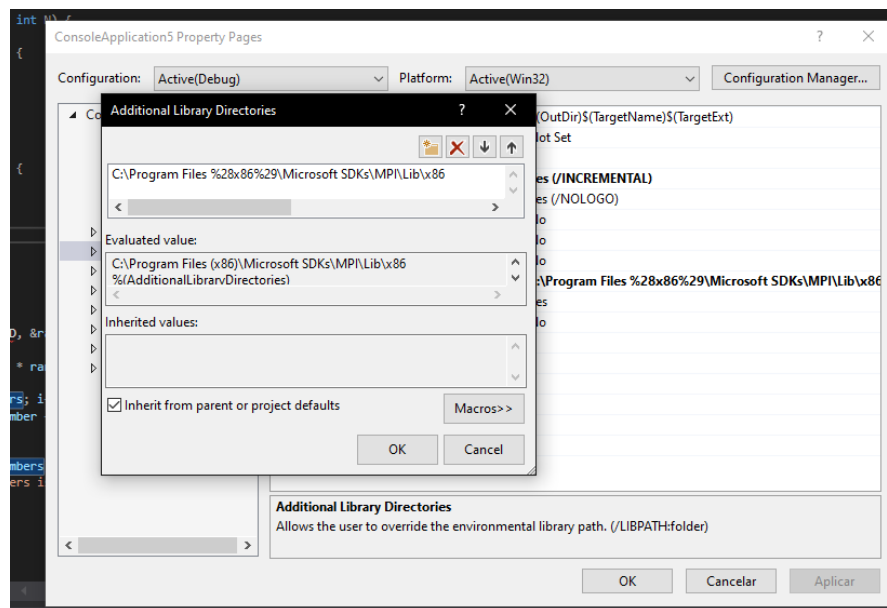


Figura 13 – Diretório adicional MPI lib

Após a última inclusão, os problemas efetivamente desapareceram, no sentido em que o `#include <mpi.h>` já não dava erros. No entanto, na linha de comandos CMD, onde supostamente tínhamos a tarefa de executar o programa com o seguinte comando, “`mpiexecute -n (nr. de processos) ./nomedoprograma`”, o sistema não reconhecia o comando, ou qualquer comando de executar os programas em MPI.

Optou-se por pesquisar mais arduamente, a fim de solucionar o problema. Encontrou-se uma resposta, no sentido em que, para podermos usufruir do MPI no Windows, no nosso caso em específico, dada a natureza do sistema operativo, a solução o era reverter a versão o do MPI, e ao invés de se utilizar a que estava efetivamente no site da Microsoft, fez-se o download da mais antiga, e surpreendente mais estável, MPICH2.

O seu processo de instalação era ligeiramente mais complexo, menos straightforward, uma vez que era necessário cumprir com uma série de passos, para se poder usar os comandos do MPI através da linha de comandos.

Para tal o primeiro passo, após a instalação do MPICH2, foi ir as variáveis de ambiente de Windows e adicionar o diretório ao PATH.

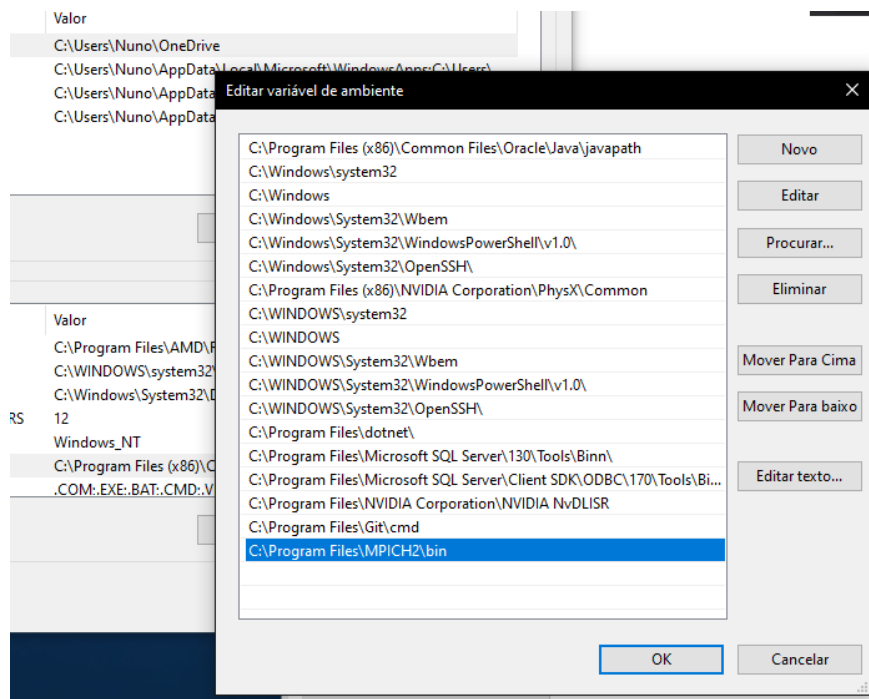


Figura 14 – Diretório adicional MPI bin no PATH do windows

Feito isto já podemos utilizar com efetividade a biblioteca na linha de comandos, como comprovado na figura abaixo.

```

Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Nuno>mpiexec

Usage:
mpiexec -n <maxprocs> [options] executable [args ...]
mpiexec [options] executable [args ...] : [options] exe [args] : ...
mpiexec -configfile <configfile>

options:
standard:
-n <maxprocs>
-wdir <working directory>
-configfile <filename> -
    each line contains a complete set of mpiexec options
    including the executable and arguments
-host <hostname>
-path <search path for executable, ; separated>
  
```

Figura 15 - mpiexec

Feito isto, foi apenas necessário registar uma nova senha e nome de utilizador ao MPI, através da linha de comandos e perguntar depois ao sistema se o mesmo é válido.

```
C:\Users\Nuno>mpiexec -validate
```

Figura 16 – Validar registo

Posteriormente a isto, foi retornada a mensagem sucesso, ou seja, efetivamente já temos acesso a utilizar o comando para executar as nossas aplicações.

Como podemos verificar na figura abaixo, o comando funciona.

```
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Nuno\Desktop\Programação Paralela\Projeto Final\soma4nrs\bin\Debug>mpiexec -n 4 ./soma5nrs
User credentials needed to launch processes:
account (domain\user) [DESKTOP-6FONSLO\Nuno]: DESKTOP-6FONSLO\Nuno
password:
launch failed: CreateProcess(soma5nrs) on 'DESKTOP-6FONSLO' failed, error 2 - O sistema não conseguiu localizar o ficheiro especificado.
launch failed: CreateProcess(soma5nrs) on 'DESKTOP-6FONSLO' failed, error 2 - O sistema não conseguiu localizar o ficheiro especificado.
launch failed: CreateProcess(soma5nrs) on 'DESKTOP-6FONSLO' failed, error 2 - O sistema não conseguiu localizar o ficheiro especificado.
launch failed: CreateProcess(soma5nrs) on 'DESKTOP-6FONSLO' failed, error 2 - O sistema não conseguiu localizar o ficheiro especificado.

Error posting writev, Uma ligação estabelecida foi anulada pelo software no computador anfitrião.(10053)
unable to post a write for the next command,
sock error: Error = 10053

unable to post a write of the close command to tear down the job tree as part of the abort process.
unable to post an abort command.

C:\Users\Nuno\Desktop\Programação Paralela\Projeto Final\soma4nrs\bin\Debug>mpiexec -n 4 ./soma4nrs
The sum of the numbers is 51.000000
The max of the numbers is 11.000000
The sum of the numbers is 87.000000
The max of the numbers is 17.000000
The sum of the numbers is 15.000000
The max of the numbers is 5.000000
The sum of the numbers is 123.000000
The max of the numbers is 23.000000

C:\Users\Nuno\Desktop\Programação Paralela\Projeto Final\soma4nrs\bin\Debug>
```

Figura 17 – Programa soma 6 primeiros números

A descrição do nome do programa é soma e máximo dos 4 primeiros números, e se virmos nos processos, foram efetivamente retornados os 4 processos.

O próximo algoritmo é acerca da soma de 100 números.

```
The data to sum : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Node 0 has numbers to sum : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Node 0 computes the sum 325
Node 1 has numbers to sum : 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
Node 1 computes the sum 950
Node 2 has numbers to sum : 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75
Node 2 computes the sum 1575
Node 3 has numbers to sum : 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
Node 3 computes the sum 2200
```

Figura 18 – Soma 100 números

O programa faz efetivamente o seu propósito, podemos confirmar pelo número de nós no mesmo, no entanto, devido a natureza do ambiente de trabalho ser Windows, alguns problemas de incompatibilidade subsistiram. Não ficam numerados corretamente.

```
Fatal error in PMPI_Gather: Invalid buffer pointer, error stack:
PMPI_Gather(863): MPI_Gather(sbuf=000000000061FD90, scount=1, MPI_INT, rbuf=000000000061FD90, rcount=1, MPI_INT, root=0,
MPI_COMM_WORLD) failed
PMPI_Gather(806): Buffers must not be aliased
```

Figura 19 – Erro Gather()

Tentou-se diagnosticar o problema e chegou-se a conclusão de que, o problema era mais uma vez a incompatibilidade do sistema operativo com algumas funções do MPI. Nomeadamente o originário, é o MPI_Gather. Crê-se que a solução era utilizar MPI_Gatherv, mas mesmo assim o problema persiste.

Por último foi testada uma multiplicação entre duas matrizes, em que $a \times b = c$.

```
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Nuno\Desktop\Programação Paralela\Projeto Final\multimatrices\bin\Debug>mpiexec -n 1 ./multimatrices

localA=
  0.0    1.0    2.0
  1.0    2.0    3.0
  2.0    3.0    4.0

localB=
  0.0   -1.0   -2.0
  1.0    0.0   -1.0
  2.0    1.0    0.0

localA=
  0.0    1.0    2.0
  1.0    2.0    3.0
  2.0    3.0    4.0

localB=
  0.0   -1.0   -2.0
  1.0    0.0   -1.0
  2.0    1.0    0.0

localC=
  5.0    2.0   -1.0
  8.0    2.0   -4.0
 11.0    2.0   -7.0

globalC =
  5.0    2.0   -1.0
  8.0    2.0   -4.0
 11.0    2.0   -7.0

C:\Users\Nuno\Desktop\Programação Paralela\Projeto Final\multimatrices\bin\Debug>
```

Figura 20 – Multiplicação de duas matrizes

Serão disponibilizados os algoritmos no diretório deste trabalho 6.

13. CONCLUSÕES E OBSERVAÇÕES

Crê-se que o objetivo para este último trabalho foi alcançado, no sentido em que, pudemos experimentar entre várias versões da biblioteca do MPI, e quanto a execução de algoritmos utilizando as funções, constantes, e diretrizes existentes na documentação do MPI.

Foi um trajeto com uma curva de aprendizagem acentuada, no sentido em que era imprescindível consultar constantemente a documentação, para nos podermos situar no que estávamos a fazer, e certamente crê-se que esse objetivo em particular foi alcançado. Aliado a estes conhecimentos adquiridos via documentação disponível da web, reiteramos também que assistir as aulas do nosso docente professor Lobinho, foi igualmente bastante esclarecedora, quando a natureza do tema debatido nestas últimas aulas, MPI, como na paralelização de algoritmos no geral.

Sem qualquer dúvida, que conseguimos compreender melhor agora no potencial em se usar a paralelização em algoritmos desenvolvidos, onde na maior parte dos casos a paralelização implica diminuição dos tempos de execução de algoritmos, que mais tarde se pode traduzir também em custos reduzidos naturalmente.

Quanto ao processo, neste caso a curva de aprendizagem, a verdade é que existiram muitas condicionantes, que foram obviamente solucionadas, no entanto mesmo com a documentação extensiva online, alguns erros com que nos deparamos não existe grande esclarecimento da documentação do MPI. Nomeadamente um deles, neste caso problemas

de incompatibilidade, em que certas funções do MPI, como por exemplo MPI_Gather gera problemas na compilação do algoritmo em Windows, ao passo que em Linux, não existe qualquer problema. Leva-nos a crer que quase toda a documentação em MPI foi desenvolvida com especial detalhe aos utilizadores de Linux.

Resta-nos, no futuro, assim que pudermos continuar a nossa aprendizagem com a paralelização de algoritmos, recorrer a um sistema operativo, mas propicio para essa tarefa. Onde exista maior documentação e suporte.

14. REFERENCIAS WEB GRÁFICAS

[1] - MPICH User's Guide. Disponível em:

<<https://www.mpich.org/static/downloads/3.4.1/mpich-3.4.1-userguide.pdf>> |

Acesso em: 5 de maio de 2021.

[2] - MPI Tutorial Introduction. Disponível em:

<<https://mpitutorial.com/tutorials/mpi-introduction>> | Acesso em: 7 de maio de 2021.

[3] - MPI Kick Start. Disponível em:

<https://www.cmpe.boun.edu.tr/sites/default/files/mpi_install_tutorial.pdf> | Acesso em: 7 de maio de 2021.

[4] - High-Performance Portable MPI. Disponível em: <<https://www.mpich.org>> |

Acesso em: 5 de maio de 2021.