CAP-372 / 2019 - QUARTA LISTA DE EXERCÍCIOS

Aluno: xxxxxxx xxxxxxx xxxxxxx

Data: 25/08/19

Exercício de integração numérica. Função escolhida:

$$\int x^n dx = \frac{1}{n+1} x^{n+1}$$

Intervalo = [a,b] = [-100 , 900]

O resultado é = 164.000.000.000

Programa:

```
! $ gfortran -o intetrap intetrap.f90
! $ ./intetrap
PROGRAM intetrap
   IMPLICIT NONE
   DOUBLE PRECISION, PARAMETER ::  a = -100D0, b = 900D0
   CALL CalcInterv(a, b, 2**10)
   CALL CalcInterv(a, b, 2**20)
   CALL CalcInterv(a, b, 2**30)
CONTAINS
   FUNCTION F(x)
      IMPLICIT NONE
      DOUBLE PRECISION :: F
      DOUBLE PRECISION, INTENT(IN) :: x
      F = (x**3)  ! function x^3
   END FUNCTION F
   FUNCTION TRAP(local_a, local_b, local_n)
      IMPLICIT NONE
      DOUBLE PRECISION :: TRAP
      DOUBLE PRECISION, INTENT(IN) :: local_a, local_b
      INTEGER, INTENT(IN) :: local_n
      INTEGER :: i
      DOUBLE PRECISION :: integral, x, h
      integral = (F(local_a) + F(local_b))/2.0
      x = local_a
      h = (local_b - local_a)/local_n
      DO i = 1, local_n - 1
         x = x + h
         integral = integral + F(x)
      ENDDO
      TRAP = integral*h
   END FUNCTION TRAP
   SUBROUTINE CalcInterv(a, b, n)
      IMPLICIT NONE
      DOUBLE PRECISION, INTENT(IN) :: a, b
      INTEGER, INTENT(IN) :: n
      DOUBLE PRECISION :: r
      REAL :: t1, t2
      CALL CPU_TIME(t1)
      r = TRAP(a, b, n)
      CALL CPU_TIME(t2)
      PRINT *,"Result=", r,"for", n, "partitions in", t2-t1, "s"
   END SUBROUTINE CalcInterv
END PROGRAM intetrap
```

Teste no PC local (Intel i7):

```
$ gfortran -o intetrap intetrap.f90
$ ./intetrap
 Result=   164000190734.86325      for        1024 partitions in   3.89998313E-05 s
 Result=   164000000000.18250      for     1048576 partitions in   2.44660005E-02 s
 Result=   164000000000.00443      for  1073741824 partitions in   4.90481424     s
```

Usando MPI:

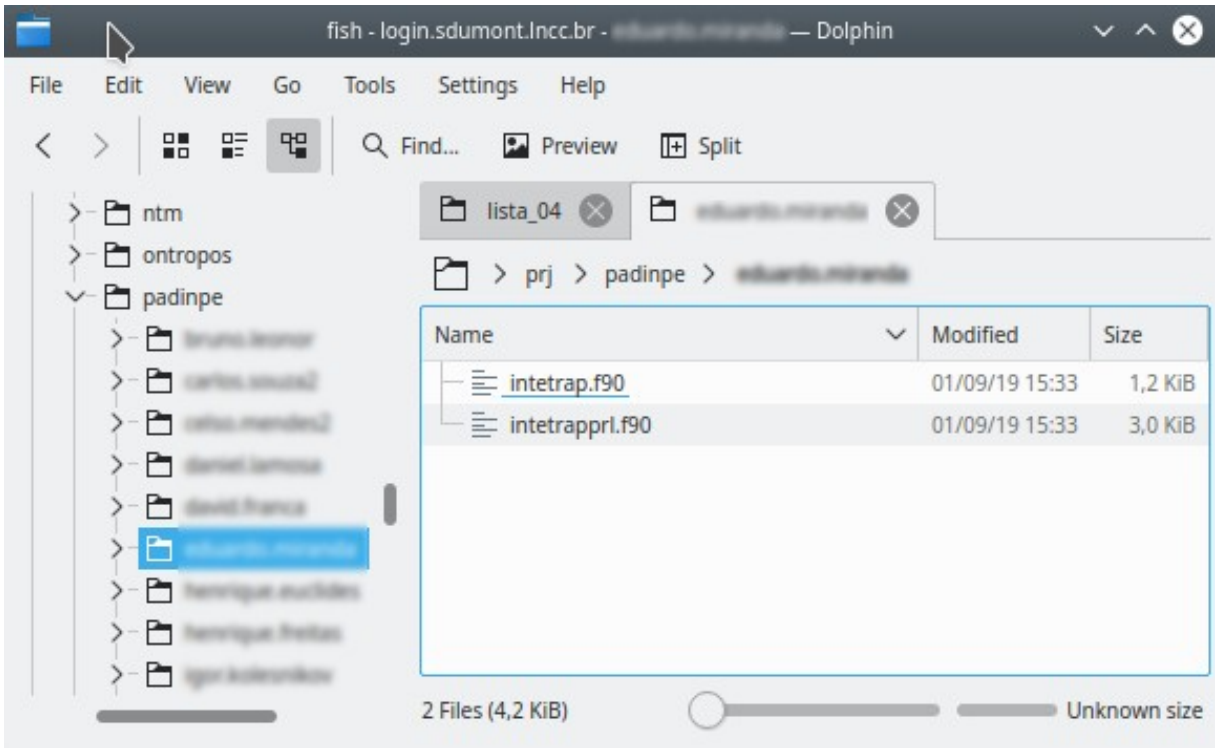
```
! CAP372 exercise 4 - trapezoidal rule - 2019-09-01
! function: y = x³ ; integral: x⁴ / 4
! range a = -100 and b = 900 , result = 164.000.000.000
! cat /proc/cpuinfo | grep proc  --> 4
! $ mpif90 -o intetrapprl intetrapprl.f90
! $ mpirun -n 4 ./intetrapprl
PROGRAM intetrapprl
  USE MPI
  IMPLICIT NONE
  INTEGER :: n    ! qty of partitions; should be multiple of p
  INTEGER :: ierror
  DOUBLE PRECISION, PARAMETER :: a = -100D0, b = 900D0  ! interval

  CALL MPI_Init(ierror)

  n = 2**10
  CALL CalcInterv(a, b, n)
  n = 2**20
  CALL CalcInterv(a, b, n)
  n = 2**30
  CALL CalcInterv(a, b, n)

  CALL MPI_Finalize(ierror)
CONTAINS
  FUNCTION f(x)
    IMPLICIT NONE
    DOUBLE PRECISION :: f
    DOUBLE PRECISION, INTENT(IN) :: x

    f = (x**3)    ! function x³
  END FUNCTION F

  FUNCTION Trap(local_a, local_b, local_n)
    IMPLICIT NONE
    DOUBLE PRECISION :: Trap
    DOUBLE PRECISION, INTENT(IN) :: local_a, local_b
    INTEGER, INTENT(IN) :: local_n
    INTEGER :: i
    DOUBLE PRECISION :: integral, x, local_h

    integral = (f(local_a) + f(local_b))/2.0
    x = local_a
    local_h = (local_b - local_a)/local_n
    DO i = 1, local_n - 1
      x = x + local_h
      integral = integral + f(x)
    ENDDO
    Trap = integral * local_h
  END FUNCTION Trap

  SUBROUTINE CalcInterv(a, b, n)
    IMPLICIT NONE
    DOUBLE PRECISION, INTENT(IN) :: a, b  ! interval
    INTEGER, INTENT(IN) :: n              ! qty of partitions
    INTEGER :: my_rank                    ! rank
    INTEGER :: p                          ! qty of processes
    INTEGER :: source, dest, ierror, part_n
    INTEGER, PARAMETER :: tag = 0
    INTEGER, DIMENSION(MPI_STATUS_SIZE) :: status
    DOUBLE PRECISION :: r, tot, h, hn, part_a, part_b
    REAL :: ta, tb, tc

    CALL MPI_Comm_rank(MPI_COMM_WORLD, my_rank, ierror)
    CALL MPI_Comm_size(MPI_COMM_WORLD, p, ierror)
    CALL MPI_Barrier(MPI_COMM_WORLD, ierror)
    CALL CPU_TIME(ta)

    h = (b - a) / n                     ! partition size
    part_n = n / p                      ! partitions per process
    hn = h * part_n                     ! process size
```

```
    part_a = a + my_rank * hn           ! partition start
    part_b = part_a + hn                ! partition end

    r = Trap(part_a, part_b, part_n)    ! partition area calculation

    IF(my_rank .EQ. 0) THEN
      tot = r
      DO source = 1, p - 1
        CALL MPI_Recv(r, 1, MPI_DOUBLE_PRECISION, source, tag, MPI_COMM_WORLD, &
                      status, ierror)
        tot = tot + r
      END DO
      CALL CPU_TIME(tb)
      tc = tb - ta
      PRINT *,"Result=", tot,"for", n, "partitions in", tc, "s"
!      PRINT *,"Result=", tot
!      PRINT *,"Partitions:", n
!      PRINT *,"Partition size:", h
!      PRINT *,"Partitions per process:", part_n
!      PRINT *,"Process size:", hn
!      PRINT *,"Elapsed:", tc, "s"
    ELSE  ! my_rank .NE. 0
      dest = 0;
      CALL MPI_Send(r, 1, MPI_DOUBLE_PRECISION, dest, tag, MPI_COMM_WORLD, ierror)
    END IF
  END SUBROUTINE CalcInterv
END PROGRAM intetrapprl
```

Teste no PC local (Intel i7):

```
$ mpif90 -o intetrapprl intetrapprl.f90
$ mpirun -n 4 ./intetrapprl
 Result=   164000190734.86325      for        1024 partitions in   5.50001860E-05 s
 Result=   164000000000.18051      for     1048576 partitions in   7.85399973E-03 s
 Result=   164000000000.00674      for  1073741824 partitions in   2.58868909     s
```

Copiando os fontes para /prj



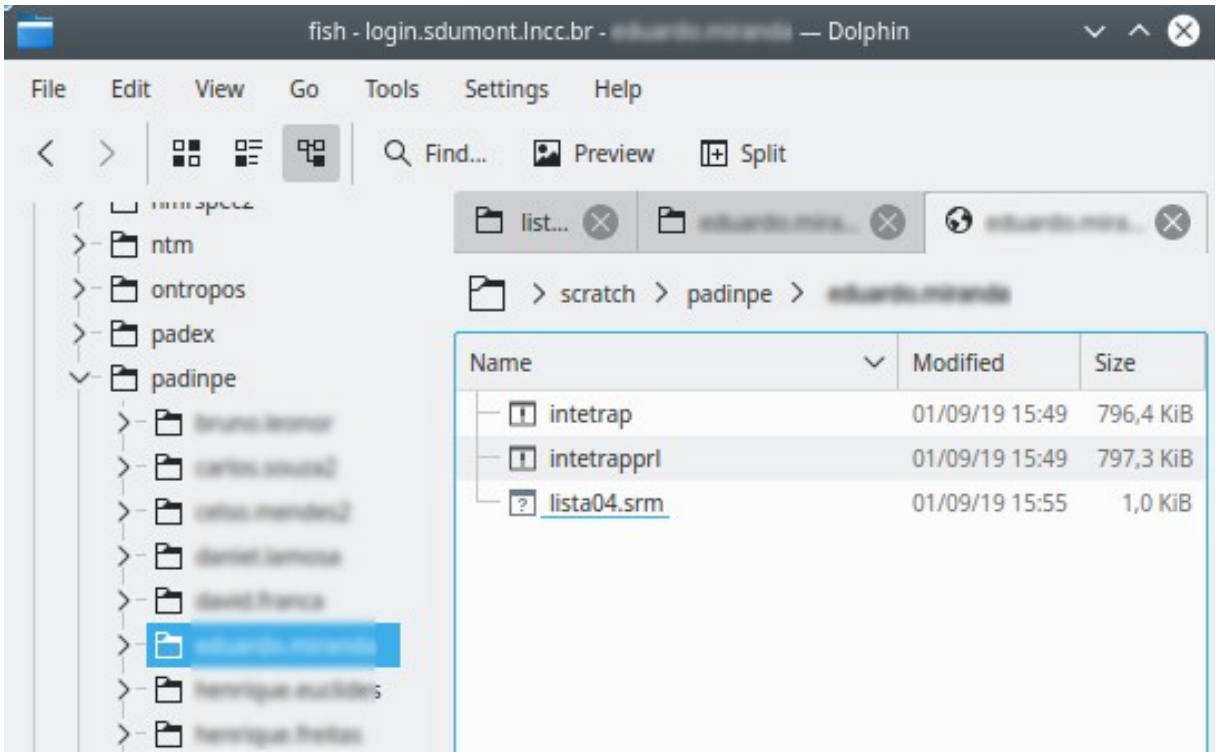
Compilando em /prj

```
$ ssh xxxxxxx.xxxxxxx@login.sdumont.lncc.br
xxxxxxx.xxxxxxx@login.sdumont.lncc.br's password:
Last login: Sun Aug 25 10:02:35 2019 from 146.134.221.157
_____________________________________________________________________
  _____ _____                                 _
 / ____|  __ \                               | |
| (___ | |  | |_   _ _ __ ___   ___  _ __ | |_
 \___ \| |  | | | | | '_ ` _ \ / _ \| '_ \| __|
 ____) | |__| | |_| | | | | | | (_) | | | | |_
|_____/|_____/ \__,_|_| |_| |_|\___/|_| |_|\__|



Manual: http://sdumont.lncc.br/support_manual.php
_____________________________________________________________________

The available softwares can be listed with the command: module avail
If there's something missing, please get in contact with helpdesk-sdumont@lncc.br
_____________________________________________________________________

[xxxxxxx.xxxxxxx@sdumont11 ~]$ module load intel_psxe/2019
[xxxxxxx.xxxxxxx@sdumont11 ~]$ mpiifort -o intetrap intetrap.f90
[xxxxxxx.xxxxxxx@sdumont11 ~]$ mpiifort -o intetrapprl intetrapprl.f90
[xxxxxxx.xxxxxxx@sdumont11 ~]$
```

Execução para testes em /prj

```
[xxxxxxx.xxxxxxx@sdumont11 ~]$ ./intetrap
 Result=   164000190734.863      for        1024 partitions in  3.0000228E-06 s
 Result=   164000000000.182      for     1048576 partitions in  5.9999991E-04 s
 Result=   164000000000.014      for  1073741824 partitions in  0.5608560     s
[xxxxxxx.xxxxxxx@sdumont11 ~]$ ./intetrapprl
 Result=   164000190734.863      for        1024 partitions in  3.9972365E-06 s
 Result=   164000000000.182      for     1048576 partitions in  6.5699965E-04 s
 Result=   164000000000.014      for  1073741824 partitions in  0.6175130     s
[xxxxxxx.xxxxxxx@sdumont11 ~]$
```

Copiando os executáveis para /scratch



Rodando em /scratch

```
[xxxxxxx.xxxxxxx@sdumont11 ~]$ cd /scratch/padinpe/xxxxxxx.xxxxxxx
[xxxxxxx.xxxxxxx@sdumont11 xxxxxxx.xxxxxxx]$ sbatch lista04.srm
Submitted batch job 395819
[xxxxxxx.xxxxxxx@sdumont11 xxxxxxx.xxxxxxx]$ ls
intetrap  intetrapprl  lista04.srm  slurm-395819.out
[xxxxxxx.xxxxxxx@sdumont11 xxxxxxx.xxxxxxx]$
```

SAÍDA

```
$ cat slurm-395819.out
sdumont1391
sdumont1391
 Result=   164000190734.863      for        1024 partitions in  4.0000305E-06 s
 Result=   164000190734.863      for        1024 partitions in  4.9998052E-06 s
 Result=   164000190734.863      for        1024 partitions in  1.1000317E-05 s
 Result=   164000190734.863      for        1024 partitions in  1.2999866E-05 s
 Result=   164000190734.863      for        1024 partitions in  4.0000305E-06 s
 Result=   164000190734.863      for        1024 partitions in  5.0002709E-06 s
 Result=   164000190734.863      for        1024 partitions in  5.0002709E-06 s
 Result=   164000190734.863      for        1024 partitions in  3.9995648E-06 s
 Result=   164000000000.182      for     1048576 partitions in  6.0799997E-04 s
 Result=   164000000000.182      for     1048576 partitions in  6.0799997E-04 s
 Result=   164000000000.182      for     1048576 partitions in  6.0700066E-04 s
 Result=   164000000000.182      for     1048576 partitions in  6.0500018E-04 s
 Result=   164000000000.182      for     1048576 partitions in  5.7100039E-04 s
 Result=   164000000000.182      for     1048576 partitions in  5.7100039E-04 s
 Result=   164000000000.182      for     1048576 partitions in  6.6399947E-04 s
 Result=   164000000000.182      for     1048576 partitions in  6.2000006E-04 s
 Result=   164000000000.014      for  1073741824 partitions in  0.6153200     s
 Result=   164000000000.014      for  1073741824 partitions in  0.6169440     s
 Result=   164000000000.014      for  1073741824 partitions in  0.6181700     s
 Result=   164000000000.014      for  1073741824 partitions in  0.6196440     s
 Result=   164000000000.014      for  1073741824 partitions in  0.6197030     s
 Result=   164000000000.014      for  1073741824 partitions in  0.6206610     s
 Result=   164000000000.014      for  1073741824 partitions in  0.6207520     s
 Result=   164000000000.014      for  1073741824 partitions in  0.6231370     s
 Result=   164000190734.863      for        1024 partitions in  3.4260005E-03 s
```

```
Result=   164000000000.182      for     1048576 partitions in  1.0200590E-04 s
Result=   163999999999.998      for  1073741824 partitions in  8.8807002E-02 s
```

A saída não foi o esperado, ficou diferente dos demais testes realizados no pc local e em /prj . Porém é possível utilizar os tempos medidos.

RESULTADO

O Speedup é o tempo de um programa serial dividido pelo tempo do programa paralelo:

Para 2^{10} partições S1 = 3.4260005E-03 / 5.0002709E-06 = 685

Para 2^{20} partições S2 = 1.0200590E-04 / 6.6399947E-04 = 0,15

Para 2^{30} partições S3 = 8.8807002E-02 / 0.6231370 = 0,14

A Eficiência é o Speedup dividido pelo nro. de processadores:

E1 = 685 / 8 = 85

E2 = 0,019

E3 = 0,018

CONCLUSÃO

Foi possível rodar conforme descrito na lista de exercícios, e os resultados mostraram que para um grande número de partições (áreas) o resultado em /scratch foi pior do que rodando em /prj . É necessário investigar mais para ter certeza, porém aparentemente isso mostra que existe uma parte não paralelizável que está influenciando no resultado. Também é necessário investigar por que a tela de saída ficou diferente em /scratch quando comparado com a execução em /prj e no pc local.

REFERÊNCIAS

- http://www.lac.inpe.br/~stephan/
- http://en.wikipedia.org/wiki/Trapezoidal_rule
- http://stackovervflow.com
- http;//fortranwiki.org
- https://www.open-mpi.org
- https://computing.llnl.gov
- https://archive.org/details/ParallelProgrammingWithMPI
- https://www.cs.usfca.edu/~peter/ppmpi/
- https://www.codingame.com/playgrounds/349/introduction-to-mpi/
- https://www.mpich.org
- https://mpitutorial.com
- https://slurm.schedmd.com