CAP-372 / 2019 - TERCEIRA LISTA DE EXERCÍCIOS

Aluno: xxxxxxx xxxxxxx xxxxxxx

Data: 25/08/19

MULTIPLICAÇÃO DE MATRIZES

Multiplicação de matrizes em Fortran 90 nas formulações que aparecem nos slides 44 e 45 do material da disciplina relativo ao livro HPC (High Performance Computing, Dowd & Severance, editora O'Reilly, 2a. edição, 1998). Disponível em Connexions: <http://cnx.org/content/col11136/1.5>

Programa contendo as versões simples e otimizada:

```
PROGRAM LISTA03
    IMPLICIT NONE
    INTEGER :: I, J, K, KK, N = 512
    REAL :: T1, T2
    DOUBLE PRECISION :: TEMP0, TEMP1, TEMP2, TEMP3
    DOUBLE PRECISION, DIMENSION(1:512,1:512) :: A, B, C

    ! Inicializa
    DO J = 1, N
        DO I = 1, N
            A(I,J) = 3*I + 4*J
            B(I,J) = 5*I + 6*J
            C(I,J) = 7*I + 8*J
        ENDDO
    ENDDO

    ! Multiplicação de matrizes
    CALL CPU_TIME(T1)
    DO K = 1, N
        DO J = 1, N
            DO I = 1, N
                C(I,J) = C(I,J) + A(I,K)*B(K,J)
            ENDDO
        ENDDO
    ENDDO
    CALL CPU_TIME(T2)
    PRINT*, 'Multiplicação: ', (T2-T1), 's'
    ! Verificando alguns resultados
    DO I = 1, 3
        PRINT*, C(I, 1)
    ENDDO

    ! Inicializa
    DO J = 1, N
        DO I = 1, N
            A(I,J) = 3*I + 4*J
            B(I,J) = 5*I + 6*J
            C(I,J) = 7*I + 8*J
        ENDDO
    ENDDO

    ! PASSO I: Permutar os loops mais externos
    CALL CPU_TIME(T1)
    DO J = 1, N
        DO I = 1, N
            DO K = 1, N
                C(I,J) = C(I,J) + A(I,K)*B(K,J)
            ENDDO
        ENDDO
    ENDDO
    CALL CPU_TIME(T2)
```

```
    PRINT*, 'Permutação:    ', (T2-T1), 's'
    ! Verificando alguns resultados
    DO I = 1, 3
        PRINT*, C(I, 1)
    ENDDO

    ! Inicializa
    DO J = 1, N
        DO I = 1, N
            A(I,J) = 3*I + 4*J
            B(I,J) = 5*I + 6*J
            C(I,J) = 7*I + 8*J
        ENDDO
    ENDDO

    ! PASSO II: Loop unrolling
    CALL CPU_TIME(T1)
    DO J = 1, N
        DO I = 1, N

            KK = MOD(N, 4)
            DO K = 1, KK
                C(I,J) = C(I,J) + A(I,K)*B(K,J)
            ENDDO

            TEMP0 = 0.0
            TEMP1 = 0.0
            TEMP2 = 0.0
            TEMP3 = 0.0
            DO K = 1+KK, N, 4
                TEMP0 = TEMP0 + A(I,K)*B(K,J)
                TEMP1 = TEMP1 + A(I,K+1)*B(K+1,J)
                TEMP2 = TEMP2 + A(I,K+2)*B(K+2,J)
                TEMP3 = TEMP3 + A(I,K+3)*B(K+3,J)
            ENDDO
            C(I,J) = C(I,J)+TEMP0+TEMP1+TEMP2+TEMP3
        ENDDO
    ENDDO
    CALL CPU_TIME(T2)
    PRINT*, 'Loop unrolling:', (T2-T1), 's'
    ! Verificando alguns resultados
    DO I = 1, 3
        PRINT*, C(I, 1)
    ENDDO

END PROGRAM LISTA03
```

COMPILANDO NO SDUMONT, EM /PRJ :

GFORTRAN

```
[xxxxxxx.xxxxxxx@sdumont13 ~]$ gfortran -pg lista3.f90 -o a1.out
[xxxxxxx.xxxxxxx@sdumont13 ~]$ time ./a1.out
 Multiplicação:   0.564355969     s
   902539023.00000000
   904518166.00000000
   906497309.00000000
 Permutação:       1.00596189     s
   902539023.00000000
   904518166.00000000
   906497309.00000000
 Loop unrolling:  0.570106983     s
   902539023.00000000
   904518166.00000000
   906497309.00000000

real    0m2.162s
user    0m2.148s
sys     0m0.004s
```

Para verificar se os resultados (matriz resultante) são numericamente equivalentes, as três primeiras células da primeira coluna da matriz foram impressas e se mostraram iguais entre as versões.

GPROF (GFORTRAN)

```
[xxxxxxx.xxxxxxx@sdumont13 ~]$ gprof ./a1.out
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 99.98      2.13     2.13        1     2.13     2.13  MAIN__


                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 0.47% of 2.13 seconds

index % time    self  children    called     name
                2.13    0.00       1/1           main [2]
[1]    100.0    2.13    0.00       1         MAIN__ [1]
-----------------------------------------------
                                               <spontaneous>
[2]    100.0    0.00    2.13                 main [2]
                2.13    0.00       1/1           MAIN__ [1]
-----------------------------------------------

Index by function name

   [1] MAIN__
```

IFORTH

```
[xxxxxxx.xxxxxxx@sdumont13 ~]$ ifort -pg lista3.f90 -o a2.out
[xxxxxxx.xxxxxxx@sdumont13 ~]$ time ./a2.out
 Multiplicação:   5.4400001E-02 s
   902539023.000000
   904518166.000000
   906497309.000000
 Permutação:      5.3547002E-02 s
   902539023.000000
   904518166.000000
   906497309.000000
 Loop unrolling:  0.2838080     s
   902539023.000000
   904518166.000000
   906497309.000000

real    0m0.417s
user    0m0.396s
sys     0m0.010s
```

GPROF (IFORTH)

```
[xxxxxxx.xxxxxxx@sdumont13 ~]$ gprof ./a2.out
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.00      0.38     0.38        1   380.00   380.00  MAIN__


                     Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) for 2.63% of 0.38 seconds

index % time    self  children    called     name
                0.38    0.00       1/1           main [2]
[1]    100.0    0.38    0.00       1         MAIN__ [1]
-----------------------------------------------
                                               <spontaneous>
[2]    100.0    0.00    0.38                 main [2]
                0.38    0.00       1/1           MAIN__ [1]
-----------------------------------------------

Index by function name

   [1] MAIN__
```

PGFORTRAN

```
[xxxxxxx.xxxxxxx@sdumont13 ~]$ pgfortran -pg lista3.f90 -o a3.out
[xxxxxxx.xxxxxxx@sdumont13 ~]$ time ./a3.out
 Multiplicação:    0.3289020     s
    902539023.0000000
    904518166.0000000
    906497309.0000000
 Permutação:       0.5601761     s
    902539023.0000000
    904518166.0000000
    906497309.0000000
 Loop unrolling:   0.3008819     s
    902539023.0000000
    904518166.0000000
    906497309.0000000

real    0m1.220s
user    0m1.194s
sys     0m0.014s
```

GPROF (PGFORTRAN)

```
[xxxxxxx.xxxxxxx@sdumont13 ~]$ gprof ./a3.out
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
100.00      1.24     1.24                             MAIN_
```

Comparação do GPROF, em /prj:

gfortran (a1.out)	2.13 s
iforth (a2.out)	0.38 s
pgfortran (a3.out)	1.24 s

o iforth teve a execução mais rápida.

Tempos referentes à multiplicação, CPU_TIME(), em /prj:

	gfortran	iforth	pgfortran
Multiplicação	.56	.54	.33
Permutação	1	.54	0,56
Unrolling	.57	.28	.3

No caso do iforth provavelmente o compilador otimizou da mesma forma a multiplicação e a permutação, e os tempos foram iguais. Nos demais a permutação teve um tempo maior provavelmente porque a matriz A é percorrida na linha e não na coluna:

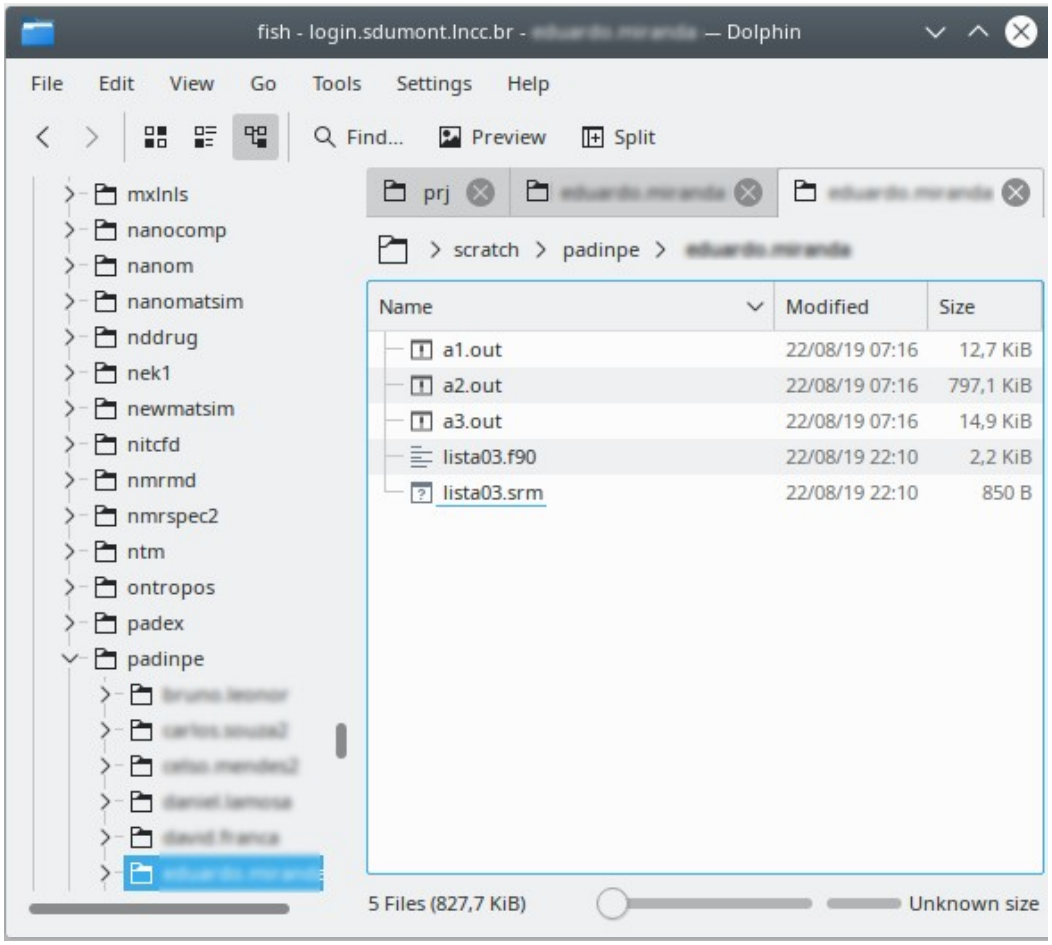
```
    DO J = 1, N
        DO I = 1, N
            DO K = 1, N
                C(I,J) = C(I,J) + A(I,K)*B(K,J)
            ENDDO
        ENDDO
    ENDDO
```

SUBMISSÃO DE JOBS

lista03.srm

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH -p cpu_dev          #Fila (partition) a ser utilizada
#SBATCH -J lista03          #Nome job
#SBATCH --time=00:05:00     #Altera o tempo limite para 5 minutos
#SBATCH --exclusive         #Utilização exclusiva do nó durante a execução do job

#Exibe os nós alocados para o Job
echo $SLURM_JOB_NODELIST
nodeset -e $SLURM_JOB_NODELIST
cd $SLURM_SUBMIT_DIR

#Configura o executavel (e analogamente para EXEC2 e EXEC3)

EXEC1=/scratch/padinpe/xxxxxxx.xxxxxxx/a1.out
EXEC2=/scratch/padinpe/xxxxxxx.xxxxxxx/a2.out
EXEC3=/scratch/padinpe/xxxxxxx.xxxxxxx/a3.out

#exibe informações sobre o executável (e analogamente para EXEC2 e EXEC3)
/usr/bin/ldd $EXEC1
/usr/bin/ldd $EXEC2
/usr/bin/ldd $EXEC3

srun -n $SLURM_NTASKS $EXEC1
srun -n $SLURM_NTASKS $EXEC2
srun -n $SLURM_NTASKS $EXEC3
```

/SCRATCH



SBATCH

```
[xxxxxxx.xxxxxxx@sdumont14 xxxxxxx.xxxxxxx]$ sbatch lista03.srm
Submitted batch job 390367
[xxxxxxx.xxxxxxx@sdumont14 xxxxxxx.xxxxxxx]$ squeue -j 390367
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
[xxxxxxx.xxxxxxx@sdumont14 xxxxxxx.xxxxxxx]$ ls
a1.out  a2.out  a3.out  gmon.out  lista03.f90  lista03.srm  slurm-390367.out
```

SACCT -j 390367

```
$ sacct --format=jobid,jobname,alloccpus,ncpus,ntasks,state,cputime,elapsed -j 390367
       JobID    JobName  AllocCPUS      NCPUS   NTasks      State    CPUTime    Elapsed
------------ ---------- ---------- ---------- -------- ---------- ---------- ----------
390367          lista03         24         24           COMPLETED   00:01:12   00:00:03
390367.batch      batch         24         24        1  COMPLETED   00:01:12   00:00:03
390367.0         a1.out          1          1        1     FAILED   00:00:00   00:00:00
390367.1         a2.out          1          1        1  COMPLETED   00:00:01   00:00:01
390367.2         a3.out          1          1        1  COMPLETED   00:00:01   00:00:01
```

Como o a1.out deu erro, para evitar consumir UA desnecessariamente, peguei do job anterior:

```
$ sacct --format=jobid,jobname,alloccpus,ncpus,ntasks,state,cputime,elapsed -j 390088
       JobID    JobName  AllocCPUS      NCPUS   NTasks      State    CPUTime    Elapsed
------------ ---------- ---------- ---------- -------- ---------- ---------- ----------
390088          lista03         24         24              FAILED   00:01:12   00:00:03
390088.batch      batch         24         24        1     FAILED   00:01:12   00:00:03
390088.0         a1.out          1          1        1  COMPLETED   00:00:03   00:00:03
390088.1         a2.out          1          1        1  COMPLETED   00:00:00   00:00:00
```

- CPUTime: Time used (Elapsed time * CPU count) by a job = NCPUS × Elapsed
- Elapsed: The jobs elapsed time

Comparação SACCT

	gfortran	iforth	pgfortran
Elapsed	00:00:03	00:00:01	00:00:01

Não consegui identificar porque estes tempos estão muito abaixo do CPU_TIME() no SLURM

Trecho extraído de SLURM-job.OUT

```
GFORTRAN
 Multiplicação:   0.580412984     s
   902539023.00000000
   904518166.00000000
   906497309.00000000
 Permutação:      0.994066000     s
   902539023.00000000
   904518166.00000000
   906497309.00000000
 Loop unrolling:  0.562052965     s
   902539023.00000000
   904518166.00000000
   906497309.00000000
IFORTH
 Multiplicação:   5.3510003E-02 s
   902539023.000000
   904518166.000000
   906497309.000000
 Permutação:      5.3543996E-02 s
   902539023.000000
   904518166.000000
   906497309.000000
 Loop unrolling:  0.2805470     s
   902539023.000000
   904518166.000000
   906497309.000000
PGFORTRAN
 Multiplicação:    0.3516920     s
    902539023.0000000
    904518166.0000000
    906497309.0000000
 Permutação:       0.5838470     s
    902539023.0000000
    904518166.0000000
    906497309.0000000
 Loop unrolling:   0.3065040     s
    902539023.0000000
    904518166.0000000
    906497309.0000000
```

CALL CPU_TIME() (saída do SLURM)

Tempos referentes à multiplicação

Para cada compilador, a 1ª coluna é rodando em /prj (login) e a 2ª em /scratch (job)

	gfortran		iforth		pgfortran	
Multiplicação	.56	.58	5.4	5.4	.33	.35
Permutalção	1.0	.99	5.4	5.4	.56	.58
Unrolling	.57	.56	.28	.28	.30	.31

Os tempos estão próximos, rodando na máquina de login ou submetendo o job

UNROLLING FACTOR

O programa usa um fator de 4 :

```
DO K = 1+KK, N, 4
       TEMP0 = TEMP0 + A(I,K)*B(K,J)
       TEMP1 = TEMP1 + A(I,K+1)*B(K+1,J)
       TEMP2 = TEMP2 + A(I,K+2)*B(K+2,J)
       TEMP3 = TEMP3 + A(I,K+3)*B(K+3,J)
ENDDO
```

Observando os tempos obtidos:

	gfortran		iforth		pgfortran	
Multiplicação	.56	.58	5.4	5.4	.33	.35
Permutalção	1.0	.99	5.4	5.4	.56	.58
Unrolling	.57	.56	.28	.28	.30	.31

o *unrolling* conforme implementado somente mostrou que é mais rápido no caso *iforth*, nos demais compiladores não houve grande ganho.

PC LOCAL

A título de curiosidade, rodando no meu PC local (Intel i7):

```
$ gfortran lista03.f90
$ time ./a.out
 Multiplicação:   0.531991005     s
   902539023.00000000
   904518166.00000000
   906497309.00000000
 Permutação:      0.814417958     s
   902539023.00000000
   904518166.00000000
   906497309.00000000
 Loop unrolling:  0.413638115     s
   902539023.00000000
   904518166.00000000
   906497309.00000000

real    0m2,079s
user    0m1,772s
sys     0m0,008s
```

para este exemplo a minha máquina local é mais rápida.

REFERÊNCIAS

- http://www.lac.inpe.br/~celso/ea266/serie4.doc
- http://www.ic.unicamp.br/~ducatte/mo401/1s2010/T2/100594-t2.pdf
- http://cnx.org/content/col11136/1.5
- https://cluster-in-the-cloud.readthedocs.io/en/latest/running.html
- https://slurm.schedmd.com/
- http://www.lac.inpe.br/~stephan/
- https://sdumont.lncc.br/
- https://en.wikipedia.org/wiki/Loop_unrolling