Introduction to High-Performance Computing

Exercise1

Giorgio Amati

Alessandro Ceci

Corso di dottorato in Ingegneria Aeronautica e Spaziale 2025

g.amati@cineca.it / g.amaticode@gmail.com

alessandro.ceci@uniroma1.it

Agenda

- ✓ Simple “warm-up” exercise: Matrix-Matrix Multiplication
- ✓ Complete the code
  - ■ Fortran/C
- ✓ Check the results
- ✓ Extract some Performance figure (in MFLOPs)

You can use:

- ✓ Any available HW
- ✓ Any available Compiler
- ✓ Any compiler option

Example: matrix-matrix multiplication

Simple problem: for 2 n^2 matrices we have to:

- ✓ compute n^3 products and n^3 sums
- ✓ load 2*n^2 data and to store n^2 data
  - ■ Ratio computation vs. load/store is O(n)!

```
do j = 1, n
   do k = 1, n
      do i = 1, n
         c(i,j) = c(i,j) + a(i,k)*b(k,j)
      enddo
   enddo
enddo
```

- ✓ MM multiplication are used for supercomputing rankings (top500)

How to do

✓ Clone the repository

- ■ `git clone https://github.com/gamati01/HPCLessons.git`

```
.
├── ESER1
│   ├── clean.sh
│   ├── compile.c.sh
│   ├── compile.fortran.sh
│   ├── EXERCISE1.pdf
│   ├── inc_precision.h
│   ├── mm.c
│   ├── mm.F90
│   ├── mod_tools.F90
│   └── README
├── LESSON1
│   ├── HPC-1.pdf
│   └── HPC-1-spoiler.pdf
└── README.md
```

mm.F90

- ✓ Complete the code

```
! main loop
call system("date     > time.log")
call timing(time1)
!
! write bottom here 3 nested loops....
!
             c(i,j) = c(i,j) + a(i,k)*b(k,j)

call timing(time2)
call system("date     >> time.log")
```

mm.c

- ✓ Complete the code

```
time1 = clock();
/*                           */
/* write here 3 nested loop */
/*                           */
           c[i][j] = c[i][j] + a[i][k]*b[k][j];


time2 = clock();
```

compile.fortran.sh

- ✓ simple script to compile the code
- ✓ choose the available compiler & compiler options

```
rm -rf *.o mm.x *.mod
#
…
#
# gfortran (GNU)
COMP=gfortran
OPT=
#
echo "compiling with " $COMP $OPT
#
$COMP $OPT mod_tools.F90 -c
$COMP $OPT mm.F90 -c
$COMP $OPT mod_tools.o mm.o -o mm.x
#
echo "That's all folks!!!"
```

compile.c.sh

✓ simple script to compile the code

✓ choose the available compiler & compiler options

```
rm -rf *.o mm.x *.mod
#
…
#
# gcc (GNU)
COMP=gcc
OPT=
#
echo "compiling with " $COMP $OPT
#
$COMP $OPT mm.c -c
$COMP $OPT mm.o -o mm.x
#
echo "That's all folks!!!"
```

./mm.x

- ✓ Fortran: If correctly code it should give an output like that

```
 --------------------------------------
  Matrix-Matrix Multiplication
  precision used              15
  rel. 0, naive multiplication
  Which matrix size?
1024
  Matrix size       =         1024
  Memory size (MB) =          24
 --------------------------------------
 initialization   1.1718750000000000E-002
   0.42933343543596444         0.94104850654997563       0.0000000000000000
 ----------------------------
 CPU:time for moltiplication  3.1406250000000000
 CPU:MFLOPS                   683.77588619402987
 CPU:check                    257.17893184193381
```

- ✓ size (e.g. 1024) given by standard input
- ✓ check ~ size/4

./mm.x

✓ C: If correctly code it should give an output like that

```
===============================
double precision
size 1024
===============================
Initialization
Elapsed time for initialization
Total time ----------------> 0.020759
===============================
Tme ----------------> 2.280420
Mflops ----------------> 941.705321
Check -------------> 252.884160
```

✓ size (e.g. 1024) hard-coded in the code

✓ check ~ size/4

Homework: Fill the table

Size	Fortran	C
256*256		
512*512		
1024*1024		
2048*2048		

- ✓ Compiler used:
- ✓ Compiler option used:
- ✓ HW used:

my homework: `gnu` compiler

Size	Size MB	Fortran MFlops (time)	C Mflops (time)
1024*1024	24	14’800 (0.15”)	14’300 (0.15”)
2048*2048	96	6’500 (2.64”)	6’500 (2.66”)
4096*4096	384	6’100 (22.6”)	6’200 (22.1”)
8192*8192	1536	6’000 (184”)	6’100 (180”)

✓ Compiler used: `gfortran/gcc` (rel. 11.4.0)
✓ Compiler option used: `-Ofast`
✓ HW used: AMD Ryzen 5 5625U with Radeon Graphics

my homework: `nvidia` compiler

Size	Size MB	Fortran MFlops (time)	C Mflops (time)
1024*1024	24	19’000 (0.11”)	15’000 (0.14”)
2048*2048	96	6’600 (2.69”)	6’200 (2.79”)
4096*4096	384	5’900 (23”)	5’500 (25”)
8192*8192	1536	5’800 (189”)	4940 (222”)

- ✓ Compiler used: **`nvfortran/nvc`** (rel. 23.11)
- ✓ Compiler option used: `-O3`
- ✓ HW used: AMD Ryzen 5 5625U with Radeon Graphics

my homework: `intel` compiler

Size	Size MB	Fortran MFlops (time)	C Mflops (time)
1024*1024	24	8’000 (0.26”)	12’500 (0.17”)
2048*2048	96	5’000 (3.4”)	5’400 (3.15”)
4096*4096	384	4’700 (29”)	5’100 (27”)
8192*8192	1536	4’600 (239”)	4’800 (231”)

- ✓ Compiler used: **`ifx/ifc`** (rel. 2024.0.2)
- ✓ Compiler option used: **`-O3`**
- ✓ HW used: AMD Ryzen 5 5625U with Radeon Graphics

mm.F90 & mm.c

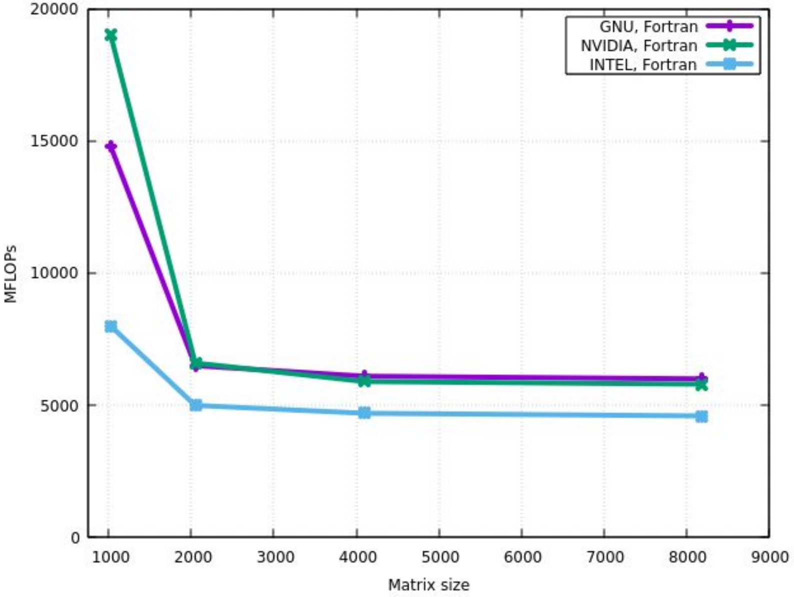
- ✓ **Fortran**

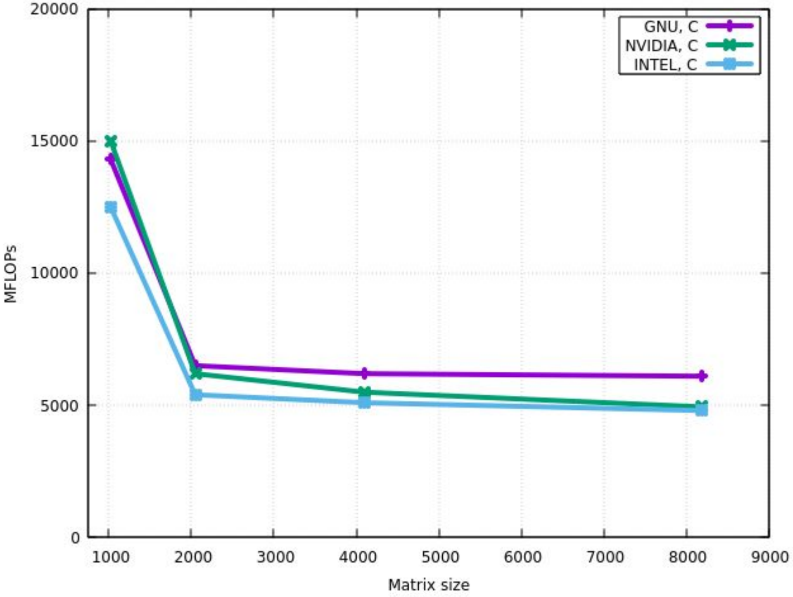
```fortran
do k = 1, n
   do j = 1, n
      do i = 1, n
         c(i,j) = c(i,j) + a(i,k)*b(k,j)
      enddo
   enddo
enddo
```

- ✓ **C**

```c
for (i = 0; i < nn; i++)
   for (k = 0; k < nn; k++)
      for (j = 0; j < nn; j++)
         c[i][j] = c[i][j] + a[i][k]*b[k][j];
```

Results





- ✓ Fortran (left) vs. C (right)
- ✓ Any idea of the reason of this behaviour?