Compilers & Optimized Librairies

- Compilers : GNU, Intel, Portland
- Memory considerations : size, top, ulimit
- Hello world ! exercice

environment

```
[bvr@hercules ~]$ module load PGI
                  pgcc -v
Export
 PGI=/opt/sw/arch/easybuild/sticky_20170712/software/PGI/16.5
 pgcc-Warning-No files to process

     module spider intel-compilers
     module load releases/2021b intel-compilers/2021.4.0
      which ifort
     module purge
     which ifort
     module load GCC
     module save mycompiler

[bvr@hercules ~]$ echo $MANPATH

                  man -k gcc
```

```
[bvr@hercules ~]$ icc -help

                Intel(R) C++ Compiler Help
                 ==========================

 Intel(R) Compiler includes compiler options that optimize for instruction
sets that are available in both Intel(R) and non-Intel microprocessors, but
may perform additional optimizations for Intel microprocessors than for
non-Intel microprocessors.  In addition, certain compiler options for
Intel(R) Compiler are reserved for Intel microprocessors.  For a detailed
description of these compiler options, including the instructions they
implicate, please refer to "Intel(R) Compiler User and Reference Guides >
Compiler Options."

 usage: icc [options] file1 [file2 ...]
        icpc [options] file1 [file2 ...]
```

SSE Streaming SIMD Extensions (1999)

SSE2 double float

SSE4 Core2 2006

AVX Advanced Vector Extensions 2008 SandyBridge

AVX2 reg. 256Bits, Haswell 2013 (AMD Zen 2017)

AVX-512 reg. 512Bits, Skylake-X 2015

AMD EPYC Naples 2017 Zen1 32 core
Roma 2019 Zen2 64 core
Milan 2021 Zen3 64 core
Genoa 2022 Zen4 96 core (AVX-512)

icc

```
[bvr@hercules ~]$ icc -help

 Optimization
 ------------
  -O0
  -O1
  -O2
  -O3
  -Ofast enable -xHOST -ipo -no-prec-div -O3 -static
                -fp-model=fast=2
 Code Generation
 ---------------
 -x<code> SSE4.2 AVX CORE-AVX512

[bvr@hercules ~]$ icc -xAVX flops.c
[bvr@hercules ~]$ ./a.out

Please verify that both the operating system and the processor support
Intel(R) X87, CMOV, MMX, FXSAVE, SSE, SSE2, SSE3, SSSE3, SSE4_1,
SSE4_2, POPCNT and AVX instructions.
```

icc

```
[bvr@hercules ~]$ icc -help

 Linking/Linker
 ------------
  -L<dir>
  -shared
  -static

   [bvr@hercules ~]$ icc -fast flops.c -o flops.fast
[bvr@hercules ~]$ icc -O0 flops.c -o flops.O0
[bvr@hercules ~]$ ls -l flops.fast flops.O0
-rwxrwxr-x 1 bvr bvr 638944  flops.fast
-rwxrwxr-x 1 bvr bvr  12725  flops.O0

   ldd  LD_LIBRARY_PATH
   file
```

```
[bvr@hercules ~]$ file flops.fast
flops.fast: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
statically linked, for GNU/Linux 2.6.18, not stripped
```

ex. flops.c

Practice yourself :

```
cp /tmp/flops.c ~
```

Compile with gcc then icc then pgcc

With and without optimization flags
(icc -xSSE2 and -xCORE-AVX512)
Use scriptgen from ceci-hpc.be
On hercules, partition batch and hmem
(or on lemaitre4 AMD EPYC Milan,
dragon2 INTEL XEON Skylake GOlD)

Memory considerations

```
cp /tmp/par40G.f ~

ifort -O0 -mcmodel=large par40G.f

size a.out
  text    data     bss        dec          hex    filename
 2698       736   43978543784  43978547218 a3d536012 a.out
bss = block started by symbol (uninitialized global data)

top

  PID USER   PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
23592 bvr    20   0   41.0g 747200   2860 R 100.0  1.1   0:04.62 a.out
```

Memory considerations

```
[bvr@hercules ~]$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) 4647764
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 515034
max locked memory       (kbytes, -l) unlimited
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 1024
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Memory considerations

```
   [bvr@dragon2 ~]$ ulimit -v 50 000 000
   [bvr@dragon2 ~]$ time ./a.out

real  1m57.198s
user  1m42.953s
sys   0m13.964s

   [bvr@dragon2 ~]$ ulimit -v 40 000 000
   [bvr@dragon2 ~]$ time ./a.out

Segmentation fault
```

Ex2. Hello world !

```
[bvr@hercules ~]$ wget
ftp://ftp.belnet.be/mirror/ftp.gnu.org/gnu/hello/hello-2.8.tar.gz
```

Untar it.
See INSTALL : ./configure; make; make install

With pgcc and icc
Use --prefix

Change the default message into the src
-> advantage of the Makefile

Librairies

```
cp -rp ~bvr/Matmul ~
```

MKL (Intel Math Kernel Lib)

BLAS - Basic Linear Algebra Subprograms
LAPACK - A package of higher level linear algebra
routines;
FFT - a set of Fast Fourier Transform routines
RNG - a set of random number generators and statistical
distribution functions.

Librairies

BLAS

http://www.netlib.org/blas/index.html

```
L1 scalar, vector and vector-vector operations
L2 matrix-vector operations
L3 matrix-matrix operations
```

Librairies

LAPACK
http://www.netlib.org/lapack/

provides routines for solving systems of simultaneous **linear equations**,
least-squares solutions of linear systems of equations,
eigenvalue problems, and singular value problems.
The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers.
Dense and banded matrices are handled, but not general sparse matrices.

for real and complex matrices, in both single and double precision.

Librairies

exercice

See matmul.F :

```
#if defined MATMUL
          c = matmul(a, b)
#elif defined BLAS
          call dgemm('no transpose','no transpose',m,n,p,
     &                1.0d0, a, m, b, n, 1.0d0, c, m)
```

With pgf90
With ifort & mkl (-qmkl)