

# Compiling

2020

Trusted partner for your Digital Journey

© Atos - For internal use

**Atos**

# Intel compiler

---

- ▶ Choosing minimum architecture:
  - `icc -xSKYLAKE` (or `-xCORE-AVX512`)
  - `icc -xCASCADELAKE`
- ▶ The extra VNNI instructions in the Cascade Lake architecture are not relevant for most HPC applications
  - However, if you have many integer computations, it may give benefits
- ▶ A good compromise for both CPUs is to use `-xCORE-AVX512`
  - Or even to try `-xCORE-AVX2`

# GCC compiler

---

- ▶ Choosing minimum architecture:
  - gcc -march=skylake
  - gcc -march=skylake-avx512
  - gcc -march=cascadelake (from GCC 9)
  
- ▶ The extra VNNI instructions in the Cascade Lake architecture are not relevant for most HPC applications
  - However, if you have many integer computations, it may give benefits
  
- ▶ A good compromise for both CPUs is to use -march=skylake-avx512
  - Or even to try -march=skylake

# Standard optimization flags

---

- ▶ -O0: reduce compilation time and make debugging work
- ▶ -O2: high optimization
- ▶ -O3: more aggressive optimization, results usually change
  - verification
  
- ▶ Intel compiler uses -O2 by default
- ▶ GCC uses -O0 by default

# Inlining

---

- ▶ If your code uses many small function calls, the overhead of a function call can be higher than the function itself. Inlining can help to remove the overhead.
- ▶ Intel compiler
  - -ip
- ▶ GCC
  - -inline

# Link-time optimization

---

- ▶ Intel compiler
  - Compiler flag `-ipo`
  - Link your applications and libraries using `xild` (dynamic libs), `xiar` (static libs)
  
- ▶ GCC compiler
  - Compiler flag `-flto`

# Debugging flags

---

- ▶ -g -O0
  - higher optimization levels will also work, but your debugger gets confused
- ▶ Intel compiler
  - -check uninit
  - -check bounds
- ▶ GCC
  - -Wall -Wextra -Wpedantic
  - -std
  - -fbounds-check (Fortran)

# Using the MKL library

---

- ▶ Intel compiler
  - -mkl=sequential, parallel, cluster
  - -static-intel might improve performance, if many calls to MKL are made (combine with -ipo and xiar)
- ▶ GCC compiler
  - Use the [Intel MKL Link line advisor](#) to find the right compile & link flags
- ▶ Use it as a drop-in replacement for the FFTW library



# Slower with more accuracy, or Faster with less accuracy

---

- ▶ -fp-model=
  - source: intermediate results are rounded, plus precise
  - precise: strict ANSI conformance
  - strict
  - consistent
  - fast
  - fast=2
  
- ▶ <http://software.intel.com/en-us/articles/consistency-of-floating-point-results-using-the-intel-compiler/>

# Using libtbbmalloc\_proxy

---

- ▶ faster allocations
- ▶ C++ code could benefit from using this library
  
- ▶ Link your application with `-ltbbmalloc_proxy`
- ▶ At runtime, set `export LD_PRELOAD=libtbbmalloc_proxy.so`
  - Library is found automatically if the intel module is loaded

# Thanks for your attention

[john.donners@atos.net](mailto:john.donners@atos.net)

Atos, the Atos logo, Atos Syntel, Unify, and Worldline are registered trademarks of the Atos group. December 2019. © 2019 Atos. Confidential information owned by Atos, to be used by the recipient only. This document, or any part of it, may not be reproduced, copied, circulated and/or distributed nor quoted without prior written approval from Atos.

The Atos logo is displayed in a bold, white, sans-serif font. The letter 'o' is stylized with a horizontal line through its center. The background of the slide features large, overlapping, semi-transparent blue circles.