Compiling Code + OpenMP + MPI

Jan Zabloudil, Moritz Siegel

CPU architectures Skylake and Cascadelake

These architectures are available in the following nodes/partitions:

- VSC-4 Login nodes l40-l49
- skylake_0096
- skylake_0384
- skylake_0768
- cascadelake_0384

You can choose between these compilers:

- gnu
- intel / oneapi / dpcpp

Usually the **intel** compilers deliver the best performance on these architectures.

CPU architectures Zen2 and Zen3

These architectures are available in the following nodes/partitions:

- VSC-5 Login nodes l50-l56
- zen3_0512
- zen3_1024
- zen3_2048
- zen3_0512_a100x2
- zen2_0256_a40x2

You can choose between these compilers:

- gnu
- intel / oneapi / dpcpp
- aocc only on **`zen`**
- nvhpc only on **`cuda-zen`**

Usually the **aocc** compiler delivers the best performance on these architectures.

Intel compilers

Intel compilers for **skylake**, **zen**, **cuda-zen**:

```
skylake trainee00@l44:~$ module avail intel/
skylake trainee00@l44:~$ module load intel/19
zen trainee00@l55:~$ module load intel/19
cuda-zen trainee00@l55$ module load intel/19
```

Intel's OneAPI compilers for **skylake**, **zen**, **cuda-zen**:

```
skylake trainee00@l44:~$ module avail compiler/
skylake trainee00@l44:~$ module load compiler/latest
zen trainee00@l55:~$ module load compiler/latest
cuda-zen trainee00@l55:~$ module load compiler/latest
```

Since both are called **icx**, type **which icx** to see which compiler is now loaded.

GNU compilers

GNU compilers for **skylake**:

```
skylake trainee00@l44:~$ module avail gcc
skylake trainee00@l44:~$ module load gcc/9.5.0-gcc-8.5.0-3wfbr74
```

GNU compilers for **zen**:

```
zen trainee00@l55:~$ module avail gcc
zen trainee00@l55:~$ module load gcc/13.2.0-gcc-12.2.0-wmf5yxk
```

GNU compilers for **cuda-zen**:

```
cuda-zen trainee00@l55:~$ module avail gcc
cuda-zen trainee00@l55:~$ module load gcc/12.2.0-gcc-9.5.0-nu5le4q
```

AOCC compilers

AOCC compilers for **skylake**:

```
None
```

AOCC compilers only for **zen**:

```
zen trainee00@l55:~$ module avail aocc
zen trainee00@l55:~$ module load aocc/4.1.0-gcc-12.2.0-rir6635
```

AOCC compilers for **cuda-zen**:

```
None
```

AOCC compilers **do not work** with NVidia's cuda package, pick **gcc** on **cuda-zen** instead.

Compiler names

Language	Intel	GNU	AOCC
C	`icx`	`gcc`	`clang`
C++	`icpx`	`g++/c++`	`clang++`
Fortran	`ifx`	`gfortran`	`flang`

Serial code examples

Examples source code location

The source files of the following examples are located at:

~training/examples/15_compiling/hello_openmp.c

Copy them to your home directory using **cp**:

```
zen traineeOO@l55:~$ cp -R ~training/examples/15_compiling/ ~
zen traineeOO@l55:~$ cd ~/15_compiling/
zen traineeOO@l55:~$ ls
```

Compiling serial C code with Intel

```
zen traineeOO@l55:~$ module load compiler/latest
```

Compile without/with optimization:

```
zen traineeOO@l55:~$ icx -OO hello.c -o hello_c
zen traineeOO@l55:~$ icx -O3 -xHost hello.c -o hello_c
```

Run the resulting binary:

```
zen traineeOO@l55:~$ ./hello_c
Hello World
```

Compiling serial C code with GNU

```
skylake trainee00@l44:~$ module load gcc/9.5.0-gcc-8.5.0-3wfbr74
zen trainee00@l55:~$ module load gcc/13.2.0-gcc-12.2.0-wmf5yxk
```

Compile without/with optimization:

```
zen trainee00@l55:~$ gcc -O0 hello.c -o hello_c
skylake trainee00@l44:~$ gcc -O2 -march=skylake hello.c -o hello_c
zen trainee00@l55:~$ gcc -O2 -march=znver3 hello.c -o hello_c
cuda-zen trainee00@l55:~$ gcc -O2 -march=znver2 hello.c -o hello_c
```

Note that only our **A40 GPU** partition on VSC-5 runs on **zen2** CPUs, all others on **zen3**!

Run the resulting binary:

```
zen trainee00@l55:~$ ./hello_c
Hello World
```

Compiling serial C code with AOCC

```
zen trainee00@l55:~$ module load aocc/3.2.0-gcc-11.2.0-y53mdzy
```

Compile without/with optimization:

```
zen trainee00@l55:~$ clang -O0 hello.c -o hello_c
zen trainee00@l55:~$ clang -O3 -march=znver3 hello.c -o hello_c
cuda-zen trainee00@l55:~$ clang -O2 -march=znver2 hello.c -o hello_c
```

Note that only our **A40 GPU** partition on VSC-5 runs on **zen2** CPUs, all others on **zen3**! Also NVidia's cuda doesn't like AOCC.

Run the resulting binary:

```
zen trainee00@l55:~$ ./hello_c
Hello World
```

Single-node parallel code with OpenMP

Compiling parallel C code with OpenMP using Intel/GNU/AOCC

```
zen trainee00@l55:~$ icx -qopenmp hello_openmp.c -o hello_openmp_c
zen trainee00@l55:~$ gcc -fopenmp hello_openmp.c -o hello_openmp_c
zen trainee00@l55:~$ clang -fopenmp hello_openmp.c -o hello_openmp_c
```

Run the resulting binary on 2 cores in parallel:

```
zen trainee00@l55:~$ export OMP_NUM_THREADS=2
zen trainee00@l55:~$ ./hello_openmp_c
Hello World... from thread = 0
Hello World... from thread = 1
```

This works identically on **skylake** and **cuda-zen** too.

Multi-node parallel code with MPI

MPI compiler wrappers

	Intel OneAPI	Intel	Openmpi
package	intel-oneapi-mpi	intel-oneapi-mpi	openmpi
compiler	`compiler/latest`	`intel/19`	any
C	`mpiicc`	`mpicc/mpigcc`	`mpicc`
C++	`mpiicpc`	`mpicxx/mpigxx`	`mpic++/mpiCC/mpicxx`
Fortran	`mpiifort`	`mpifc/mpiifort`	`mpifort`

There are many more MPI flavours available, many of them installed:

- mpich
- mvapich
- mvapich2

Find your favourite MPI flavour

List all with **spack find mpi**:

```
zen trainee00@l55:~$ spack find -l mpi
```

To find the modules you need to **grep** be specific:

```
zen trainee00@l55:~$ module avail -t | grep mpi
```

If you search for a specific mpi flavour use **module avail** directly:

```
zen trainee00@l55:~$ module avail mpi
zen trainee00@l55:~$ module avail intel-oneapi-mpi
zen trainee00@l55:~$ module avail openmpi
zen trainee00@l55:~$ module avail mvapich
```

Compare MPI compiler wrappers of Intel MPI and GNU

```
skylake trainee00@l44:~$ module purge
skylake trainee00@l44:~$ module load compiler/latest
skylake trainee00@l44:~$ module load intel-oneapi-mpi/2021.7.1-intel-2021.
skylake trainee00@l44:~$ mpiicc -show

skylake trainee00@l44:~$ module purge
skylake trainee00@l44:~$ module load gcc/12.2.0-gcc-9.5.0-aegzcbj
skylake trainee00@l44:~$ module load intel-oneapi-mpi/2021.10.0-gcc-9.5.0-
skylake trainee00@l44:~$ mpicc -show
```

Different compiler installations on different clusters

OpenMPI Compiler wrapper installations on **VSC-4**:

```
skylake trainee00@l44:~$ module avail openmpi
skylake trainee00@l44:~$ module load openmpi/4.1.4-intel-2021.7.1-yhzktow
skylake trainee00@l44:~$ module load openmpi/4.1.4-gcc-12.2.0-xt53foa
```

OpenMPI Compiler wrapper installations on **VSC-5**:

```
zen trainee00@l55:~$ module load openmpi/4.1.6-gcc-12.2.0-uk6hded
zen trainee00@l55:~$ module load openmpi/4.1.4-aocc-4.0.0-x4jtiii
cuda-zen trainee00@l55:~$ module load openmpi/4.1.4-gcc-9.5.0-rbertc2
cuda-zen trainee00@l55:~$ module load openmpi/4.1.4-nvhpc-23.3-rpvfhbj
```

Examples parallel code MPI

Compiling MPI C code with Intel and Intel MPI

```
skylake trainee00@l44:~$ module purge
skylake trainee00@l44:~$ module load compiler/latest
skylake trainee00@l44:~$ module load intel-oneapi-mpi/2021.7.1-intel-2021.
```

Compile without/with optimization:

```
skylake trainee00@l44:~$ mpiicc -O0 hello-mpi.c -o hello-mpi_c
skylake trainee00@l44:~$ mpiicc -O3 -xHost hello-mpi.c -o hello-mpi_c
```

Run the resulting binary on 2 cores in parallel:

```
skylake trainee00@l44:~$ mpirun -np 2 ./hello-mpi_c
Hello world from processor l44, rank 1 out of 2 processors
Hello world from processor l44, rank 0 out of 2 processors
```

Compiling MPI C code with GNU and Intel MPI

```
skylake trainee00@l44:~$ module purge
skylake trainee00@l44:~$ module load gcc/9.5.0-gcc-8.5.0-3wfbr74
skylake trainee00@l44:~$ module load intel-oneapi-mpi/2021.10.0-gcc-9.5.0-
```

Compile with/without optimization:

```
skylake trainee00@l44:~$ mpicc -O0 hello-mpi.c -o hello-mpi_c
skylake trainee00@l44:~$ mpicc -O2 -march=skylake hello-mpi.c -o hello-mpi
```

Run the resulting binary on 2 cores in parallel:

```
skylake trainee00@l44:~$ mpirun -np 2 ./hello-mpi_c
Hello world from processor l44, rank 1 out of 2 processors
Hello world from processor l44, rank 0 out of 2 processors
```

Examples: compiling MPI C code with AOCC and OpenMPI

```
zen traineeOO@l55:~$ module purge
zen traineeOO@l55:~$ module load aocc/4.1.0-gcc-12.2.0-rir6635
zen traineeOO@l55:~$ module load openmpi/4.1.4-aocc-4.0.0-x4jtiii
```

Compile with/without optimization:

```
zen traineeOO@l55:~$ mpicc -OO hello-mpi.c -o hello-mpi_c
zen traineeOO@l55:~$ mpicc -O2 -march=znver3 hello-mpi.c -o hello-mpi_c
```

Run the resulting binary on 2 cores in parallel:

```
zen traineeOO@l55:~$ mpirun -np 2 ./hello-mpi_c
Hello world from processor l55.vsc.xcat, rank 1 out of 2 processors
Hello world from processor l55.vsc.xcat, rank 0 out of 2 processors
```