Compiling and Running SWIFT with the icx compiler

Code

It is recommended to update to the latest version of SWIFT. In particular, a number of issues that raise warnings and/or errors only with the icx compiler were fixed on 31 January 2023.

Modules

For Cosma 8:

```
module purge
module load intel_comp/2022.3.0 compiler mpi
module load parallel_hdf5/1.14.0 parmetis/4.0.3
module load ucx/1.10.1
module load gsl/2.5
module load fftw/3.3.10cosma8
```

For Cosma 7:

```
module purge
module load intel_comp/2022.3.0 compiler mpi
module load parallel_hdf5/1.14.0 parmetis/4.0.3
module load gsl/2.5
module load fftw/3.3.10cosma7
module load ucx/1.10.1
```

For Cosma 5:

```
module purge
module load intel_comp/2022.3.0 compiler mpi
module load parallel_hdf5/1.14.0 parmetis/4.0.3
module load gsl/2.5
module load fftw/3.3.10
module load ucx/1.10.1
```

Note in particular the need to specify `fftw/3.3.10` (or c7/c8 specific variants) — older versions do not work with the `intel_comp/2022.3.0` compiler module.

Also note the need to load the ucx module, in particular for Cosma 8 — SWIFT will compile and run fine without it, but any simulations using MPI will be significantly slowed down if this is not loaded.

Beware that the initial `` `module purge` `` step does not work properly with some other compiler modules, in particular `` `intel_comp/2021.1.0 compiler` ``. If this is currently loaded, it is best to exit the terminal completely before loading the icx modules described above. The modules listed above *do* unload cleanly, if desired.

Compiling

As of 9 March 2023, the only tweak required is to add `` `--enable-compiler-warnings` `` as a flag to `` `.configure` ``. This is necessary because icx complains about a few code pieces that icc does not flag up.

Note that tests so far have been limited to the public code version with the `SWIFT-EAGLE` sub-grid suite. Other models may require additional options to work with icx.

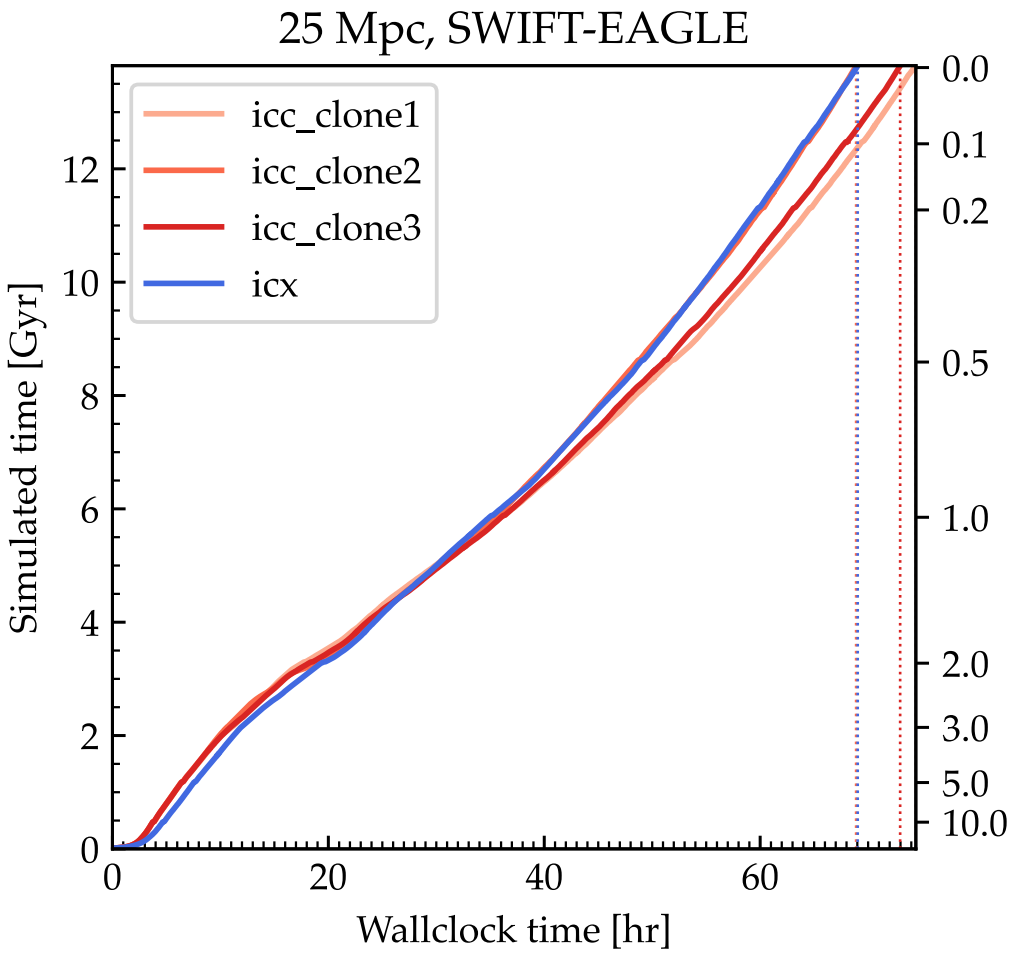
Running

Apart from loading the modules listed above, running the icx-compiled code is done in exactly the same way as for the icc version. E.g. for an MPI run over 4 nodes, with 8 ranks per node and 16 threads per rank (as recommended for Cosma 8):

```
mpirun -np 32 ./swift_mpi -v 1 \
    --cosmology --eagle \
    --threads=16 --pin --interleave \
    eagle.yml
```

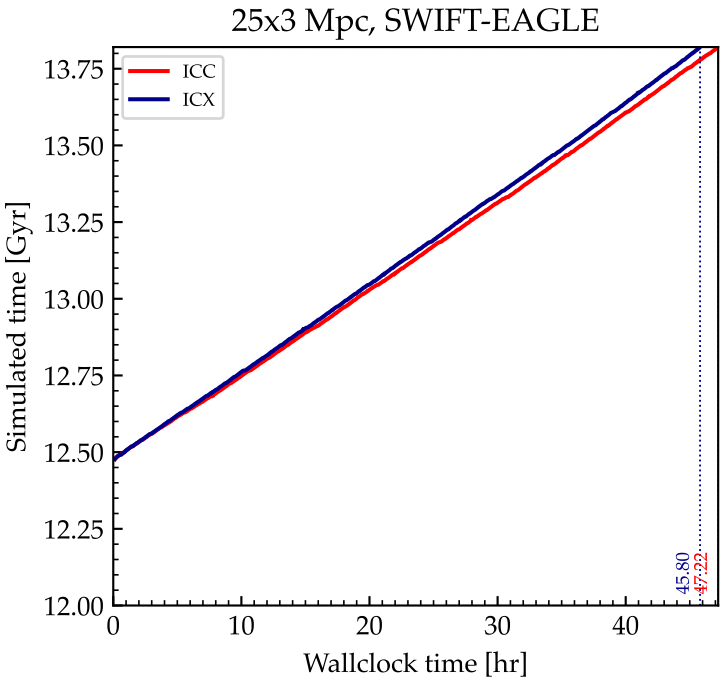
Run time comparison

On Cosma 8, there is essentially no difference in run time between icc and icx. The only exception found in tests is for very small volumes (6 Mpc, N=94^3), where icx was found to be up to ~30% slower than icc.

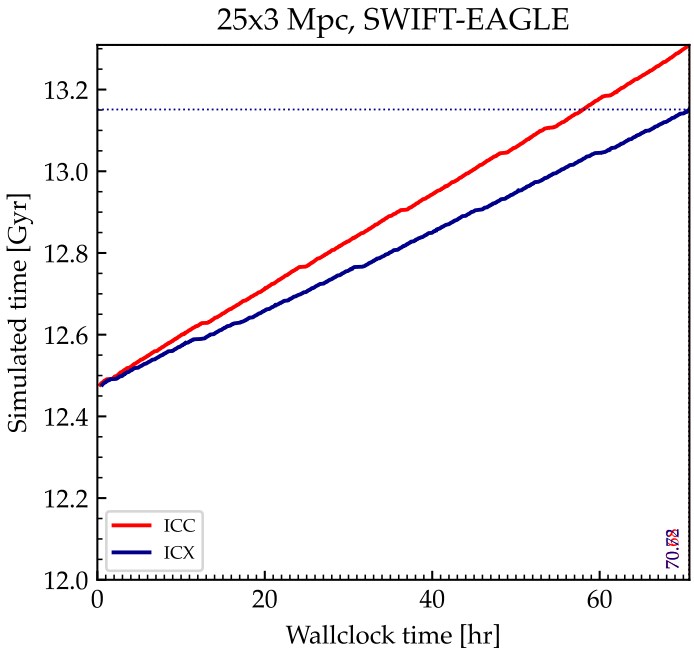


Above: run time comparison on Cosma8 for a 25 Mpc SWIFT-EAGLE simulation run with the SWIFT code on a single node (128 cores). Blue represents a run with an executable compiled with icx, the three red shades are three identical runs with an icc-compiled executable to constrain stochastic run time variations. The icx-compiled executable performs as good as, or possibly even slightly better than, icc.

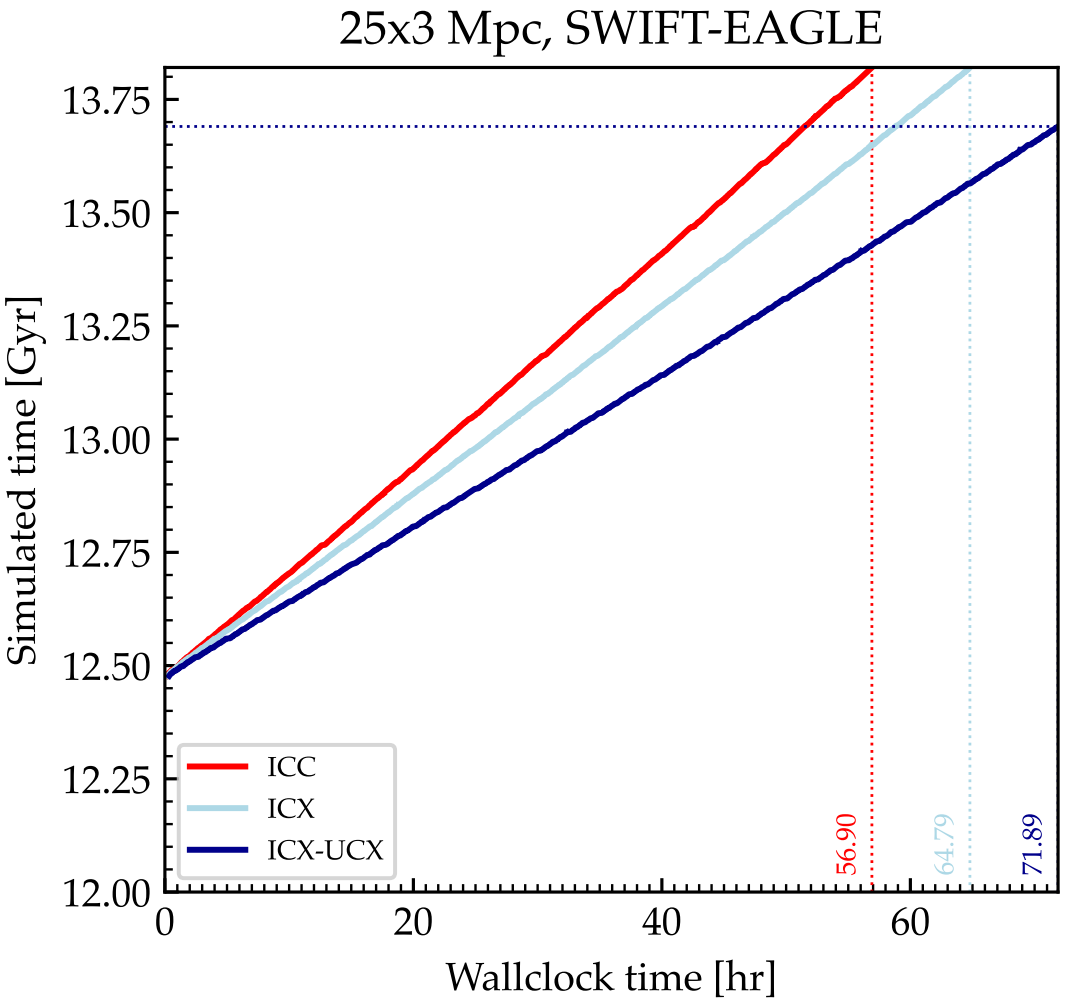
Below: run time comparison on Cosma8 for a 75 Mpc SWIFT-EAGLE simulation run with the SWIFT code on four nodes, with 8 MPI ranks per node and 16 tasks per rank (512 cores total). The test was run only from z = 0.1 to limit the computational cost. Blue represents a run with an executable compiled with icx, red an identical runs with an icc-compiled executable. Again, the icx-compiled executable performs as good as, or possibly even slightly better than, icc.



On Cosma 7, icx does seem to perform a bit worse than icc, at least for the larger simulation run over MPI. The test below is equivalent to what is shown above, but using 6 nodes of Cosma 7, with two MPI ranks per node and 14 tasks per rank (168 cores total).



On Cosma 5, the icx-compiled code also runs about 14% slower than icc, for the L=75 Mpc, N=1128^3 simulation between z = 0.1 and 0. Interestingly, loading the ucx module here appears to make the icx run even slower (under investigation). This test was performed on 32 nodes, with two MPI ranks per node and 8 tasks per MPI rank (512 cores total).



Run time comparisons including other compilers

In addition to the icx vs. icc comparisons, the performance of the 75 Mpc SWIFT-EAGLE test problem was tested with some other combinations of compilers and/or MPI implementations. All these tests were run on Cosma 7, in the same setup as above (6 nodes with 2 MPI ranks each and 14 tasks per rank). The combination of icc and Intel MPI is the one that performs best, while GCC is much slower than even icx. In general, Intel MPI runs faster than OpenMPI. Note that lines should be compared at fixed y-axis values, i.e. the difference between ICC/IntelMPI and GCC/ OpenMPI is a factor ~x4.