Performance of POP
on Snellius

20 jan 2024
Michael Kliphuis

Introduction

We tested the performance of the Parallel Ocean Program (POP) model (version pop2.1alpha_jan2005) on the dutch national supercomputer Snellius.
We tested on the so called gx1v6 grid representing a low resolution 1° grid and on the so called tx0.1v2 grid representing a high resolution 0.1° grid. This (informal) document shows the test results.

Each test was done with a simulation representing 'present day' i.e. initialized with a present day temperature and salt field (from NCAR) and forced with present day atmospheric wind speeds, surface heat fluxes and surface freshwater fluxes from the so called CORE-I climatology dataset [Large and Yeager , 2004].

Performance test results

All tests were compiled with the 2022 software stack and foss toolchain with GCC compiler which generally proves best for running on AMD cores. In order to be able to generate NetCDF output we made use of the NetCDF C and Fortran libraries and for optimization we used the OpenBLAS and Scalapack libraries. More concrete we used the following Snellius modules:

```
2022
foss/2022a
netCDF/4.9.0-gompi-2022a
netCDF-Fortran/4.6.0-gompi-2022a
OpenBLAS/0.3.20-GCC-11.3.0
ScaLAPACK/2.2.0-gompi-2022a-fb
```

Optimal settings

In order to get the most optimal setup in terms of compiler flags, type of nodes, usage of nodes, distribution of gridpoint blocks over the cores etc. we started with a high resolution 0.1° base test on 1280 'rome' cores with the default settings. We then tested new settings which we kept when there was an improvement of performance. The results are shown in the table below.

Case name	# cores	block size	settings	performance (years/24h)	comments

base	1280 rome	90x75	5 model days
			output to /projects
			
			pop_in settings:
			
			`restart_freq_opt = ‘nmonth’`
			`tavg_freq_opt  = 'nmonth'`
			`movie_freq_opt  = 'nday'`
			`movie_fmt       = 'nc'`
			`tavg_fmt_out   = 'bin'`
			`clinic(tropic)_distribution_type=cartesian`
				1.19	high resolution 0.1° base test
					
					now find settings that improve performance

ptest1	1280 rome	90x75	base test settings together with environment variables (in pop.slurm):
			
			`OMPI_MCA_fcoll="two_phase"`
			`OMPI_MCA_io_ompio_bytes_per_agg="512MB"`
				1.21	Not significantly faster, leave them out

ptest2	1152 genoa	100x75	base test settings together with: running on genoa node with 192 cores per node (instead of 128 on rome node)	1.75	Tip from Wim Rijks from SURF, genoa nodes are faster, now better performance on less cores!
ptest3	1152 genoa	100x75	ptest2 settings now for: 32 model days (instead of 5)	2.97	Initialization phase takes long so 5 days is way too short

ptest4	1152 genoa	100x75	ptest3 settings together with compile flags:
			
			`fexternal-blas –funroll-loops –flto –`
			`march=native  –lopenblas`
			`–lscalapack`
				3.13	Architecture and link time optimizations and there are many matrix multiplications so BLAS and LAPACK libraries work well

ptest5	1152 genoa	100x75	ptest4 settings but now with pop_in settings: `clinic(tropic)_distribution_type=spacecurve`	3.32	spacecurve distribution works better for high res, not for low res!

ptest6	1152 genoa	100x150	ptest5 settings but now use only half of the nodes
			
			pop_in settings:
			
			`nprocs_clinic(tropic) = 576`
			
			pop.slurm settings:
			
			`#SBATCH ––nodes 6`
			`#SBATCH ––ntasks–per–node 96`
			`#SBATCH ––cpus–per–task 2`
			`#SBATCH ––distribution=block:cyclic:block`
				4.02	Tip from Marco Verdicchio from SURF. I got SBATCH settings from Gijs van den Oord . Especially scaling is much better when using half of the nodes (see later figures)

Table 1: Settings that deliver the best performance for a high resolution 0.1° POP base test

When we started using Snellius, the performance of our POP model was very disappointing and even worse than on the previous supercomputer Cartesius. With the help of Wim Rijks and Marco Verdicchio of SURF and Gijs van den Oord of the Netherlands eScience Center we were able to improve the performance significantly. Using only half of the genoa nodes of Snellius seems to improve the scaling (with respect to Cartesius) as well.

Table 2 and figure 1 below show the test results of the low resolution POP:

Performance low resolution POP

Performance low resolution gx1v6 (1°)				
# cores	performance (modelyears/24h)	cost corehours/ 1000 modelyears	wallclock days to finish 1000 modelyears	remarks
192	128.7	35.804	7.8	Runs were all done with the so called 'cartesian' distribution. Because there are only 384 x 320 horizontal gridpoints the model does not scale very well anymore when using more than 768 cores (relatively much communication).
384	205.5	44.847	4.9	
768	285.1	64.651	3.5	
1536	307.1	120.039	3.3	

Table 2: performance low resolution POP

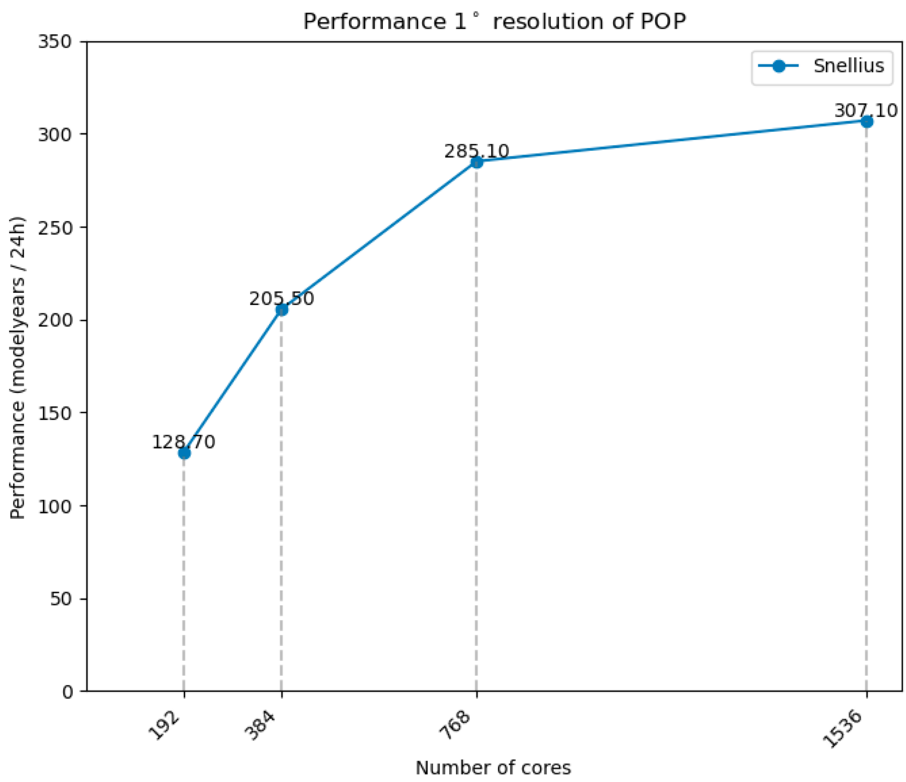

Figure 1 performance low resolution POP

Performance high resolution POP

Performance high resolution tx0.1v2 (0.1°)				
# cores	performance (modelyears/24h)	cost corehours/ 1000 modelyears	wallclock days to finish 1000 modelyears	remarks
1152	4.02	6.877.612	249	Runs were all done with the so called 'spacecurve' distribution. The model has 3600x2400 horizontal gridpoints and scales reasonable well
2304	7.64	7.237.696	131	
4608	11.41	9.692.550	88	
9216	17.06	12.965.064	59	
13824	19.42	17.084.243	52	

Table 3: performance high resolution POP

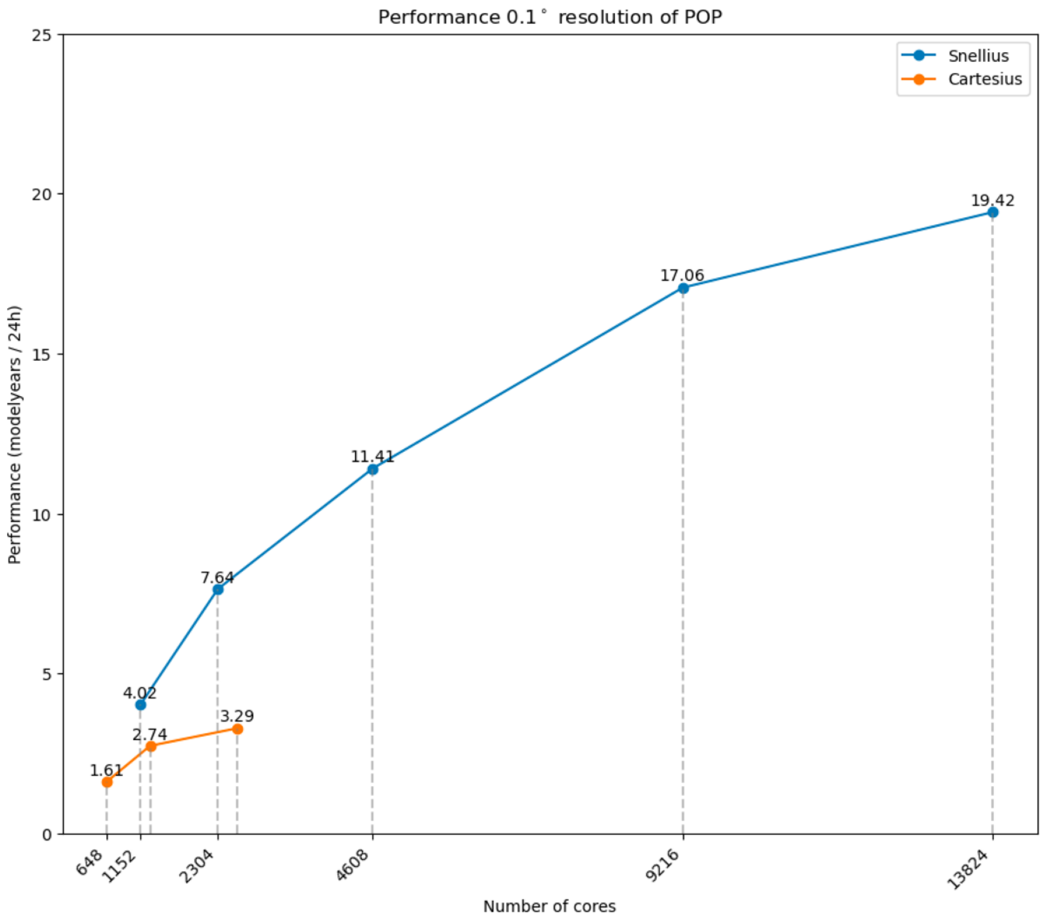

Figure 2 performance high resolution POP