Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Benchmarking and Thread Scaling of the HBM Ocean Circulation Model

Mikael Rännar[a], Maciej Szpindler[b]

[a]HPC2N & Department of Computing Science, Umeå University
[b]Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw

Abstract

The HBM (HIROMB-BOOS Model) ocean circulation model scaling on the selected PRACE Tier-0 systems is described. The model has been ported to the BlueGene/Q architecture and tested against OpenMP and mixed OpenMP/MPI parallel performance and scaling with a given test case scenario. Benchmarking of the selected computational kernels and model procedures with a micro-benchmarking module has been proposed for further integration with the model code. Details on the micro-benchmark proposal and results of the scaling tests are described.

1. Introduction

This report describes enabling of the HBM model on the PRACE Tier-0 systems. The technical work was a part of the PRACE Preparatory Access project titled “Next generation pan-European coupled Climate-Ocean Model - Phase 1 (ECOM-I)” and focused on the performance and optimization of the HBM ocean circulation model from the Danish Meteorological Institute. The scope of the report covers PRACE experts contribution involved in the project.

2. Model description

HBM (HIROMB-BOOS Model) is the ocean circulation model actively developed by the Danish Meteorological Institute (DMI). The application code is proprietary to DMI and not publicly available. Enabling work has been done within the PRACE Preparatory Access framework and under MoU agreement between DMI and respective PRACE partner institutions. The model code is written in free format standard Fortran95 and has been parallelized using OpenMP, MPI and more recently also OpenACC. For more details on modeling theory, model implementation and methods used refer to the DMI technical reports [2], [3]. Application scaling and performance has been previously tested on a number of systems. Previous enabling of the application for the selected HPC architectures has been conducted by the DMI developers team and documented in the technical report [1]. Reported work relates to the versions **`2.8`**, **`2.8-v2`** and **`2.8-green`** of the model code.

3. Scope of the work

ECOM aims to build up and optimize the computing performance of a coupled climate-ocean model for both operational marine forecasting and regional climate modeling in pan-European scale. This will provide a modeling basis for the next generation Copernicus marine and climate service (2015-2020) and IPCC AR6 evaluation. ECOM is divided into two Phases. Phase 1 (ECOM-I) focuses on the optimization of a pan-European Ocean model for operational forecasting based on HBM. The technical goals are to reduce the runtime of 10-day forecast from 16 hours (current level) to 2 hours, i.e. by 87.5%. The scalability of the pan-European HBM should go from current 900 cores to as many cores as possible. The goals will be reached by optimizing parallel computing and I/O, two-nesting as well as halo-communication in MPI. The model will be upgraded with three versions according to the improvements made.

PRACE experts working with the ECOM-I project were contributing to the benchmarking of the model code and scaling performance on the selected PRACE Tier-0 systems, particularly the Blue Gene/Q architecture.

4. Creating a micro-benchmark

One task was to create a module for saving relevant fields for two important routines, **`momeqs`** and **`tflow_int`**, both before and after the call to the routine. The files from the save before the calls are to serve as proper input sets for a micro-benchmark with the two routines. The files saved after the call are used to check that the runs with the input set actually produce the same result. The code developers demands that the output must be binary equal for every run, independent of the number of cores used.
The routine **`momeqs`** sets up and solves the momentum equations using a tri-diagonal solver. There is no OpenMP parallelization inside **`momeqs`**, but all threads run their own instance (inside an OpenMP parallel region) of **`momeqs`** for each domain.
The routine **`tflow_int`** does tracer advection and diffusion. Here, the whole routine is one OpenMP parallel region with many of function calls and barriers.
The micro-benchmarks can be used for several different purposes, e.g.,

- Experimenting with different OpenMP directives and options.
- Validate that new versions of the routines produce the same result and whether it they run faster or not.
- Testing different hardware settings to see which suit the code best.
- Run it on different hardware and maybe use the results as input for a decision about which hardware to run production on.
- Tuning settings for a specific run time environment.

The created module consists of four routines, one for saving and one for retrieving data for each of the two target routines. The save routine automatically saves all needed fields into separate files for each domain and the arguments to the routine are an id and a number to indicate in which time step to save. The id is used to be able to do several saves in a single run, e.g., saving both before and after the desired routine. We chose to use the same routine for saving both before and after, even though there are less data needed to be saved for the output than for the input.
There are some arguments to **`momeqs`** and **`tflow_int`** that have different sizes depending on the number of threads used and they are not saved. These fields are not changed during the run, but are computed in the initialization phase of the original main program, i.e., before the time steps starts. Since we like to be able to run with different number of threads, these fields will have to be computed in the initialization phase of the benchmark program as well.
The new main program for the benchmark is the old main program with no time step loop, but instead calls to the retrieve functions and then the calls to **`momeqs`** and **`tflow_int`**. Finally calls are made to save the computed fields so that one can compare the computed results with the earlier saved to make sure the results are identical. We have tried to remove as much as possible from the old main in order to reduce the overhead of the actual benchmark. There are some timers left from the old main, but one should probably insert some new more suitable to the new organization of the main program.
Testing has been done so that it is possible to serially run the old main program (with some save calls inserted) and get input and result files for the benchmark program. The input files can then be read by the new benchmark program and run with any number of threads and still produce the same output as the serial run.

5. Scaling Results

The described performance and scaling tests have been gathered on the Blue Gene/Q system FERMI in CINECA. Some specific problems with Fortran file I/O related to the XL compilers have occurred during code porting. These issues have been resolved with minor code changes related to the Fortran **`open`** and **`write`** statements and careful compile options selection.
The application has been built with the code authors guidelines included in [1] as a reference. To achieve representative and comparable results application has been built with “TUNE” [1] compilation options. For GNU compilers these options include:

```
-fopenmp -O3 -funroll-loops -ffast-math -fdump-ipa-inline
-finline-functions -finline-limit=5000
```

For the IBM XL compilers, native for Blue Gene/Q architecture, similar options have been chosen with precise settings for thread handling (OpenMP compilation):

```
-q64 -qarch=qp -g -O2 -qnoipa
-qsmp=noauto:omp:noostls:schedule=static:nospeculative:stackcheck
-qstrict=all -qmaxmem=-1
```

The model application is implemented and proved to preserve bit-reproducible results with a serial, threaded (OpenMP), parallelized (MPI) and mixed (OpenMP+MPI) builds and across different architectures. After initial difficulties managing bit-reproducibility on the Blue Gene/Q system, eventually all the problems have been fixed and the code successfully ported to the FERMI system and other smaller systems to assert versatility. Both XL and GNU compiled versions of the application preserve bit-reproducibility. Although it has been shown that the results on the Blue Gene/Q platform are not bit-wise consistent with the x86-based systems it has been agreed that the differences are related to hardware floating point implementation and are acceptable after discussion with the model developers.

OpenMP thread scaling

Having the code correctly ported to the Blue Gene/Q architecture the threaded version of the application has been tested against scaling on the one computing node. One Blue Gene/Q node has 16 CPU cores and each of these cores can execute 4 SMT (Simultaneous Multi-Threading) threads. The application has been configured for building using **`--enable-openmp`** and **`--disable-mpi`** options to eliminate possible MPI runtime overhead on thread scaling. Both XL and GNU compiled versions of the threaded-only application have been tested using 1-64 hardware threads (SMT).
For the OpenMP thread scaling runs the “MyOV3” test-case has been used. This test case constitutes the upcoming MyOcean Version 3 setup which is planned to run fully operational at DMI from 1st April 2013. The setup is modified slightly from the case presented as Variant0 in [3] in the following ways: In the Baltic Sea, the horizontal resolution is now 1 n.m. ^2, there are 122 layers with a top-layer thickness of 2 meters and a vertical resolution of 1 meter down to 100 meters depth. Initial field and bathymetry have been also improved [1].
Figure 1 shows results of the OpenMP thread scaling on the one compute node of the BlueGene/Q system. The twofold scaling curve reflects the underlying hardware configuration of the system. With the application running with 1 to 16 threads, each thread is using a physical CPU core exclusively. Further increasing the number of threads employs SMT hardware support with threads sharing the CPU resources. For this reason performance for more than 16 threads is lower but the application still benefits from larger thread count. This is visible on the scaling curve. For up to 16 threads almost linear scaling is achieved and further scaling results in the speedup of a factor 2.5 in the application runtime for 64 threads compared to 16 threads (the XL compiler case).
The thread binding layout has been also tested for a better performance using the **`BG_THREADLAYOUT`** runtime setting. The default scheme uses a breadth-first algorithm for thread placement which progresses across the cores that are defined within the process before selecting additional threads within a given core (**`BG_THREADLAYOUT=1`**). The other option is to use a depth-first algorithm that progresses within each core before switching to the next core (**`BG_THREADLAYOUT=2`**). No significant differences in time measurements have been noted with these two schemes. For complete scaling tests the default scheme has been used.

Mixed threaded parallel OpenMP+MPI scaling

The threaded parallel (OpenMP+MPI) version of the application has been configured with both **`--enable-openmp`** and **`--enable-mpi`** options. The code has been compiled with the XL compiler only because the GNU Fortran compiler shows lower performance of the model code with OpenMP enabled on Blue Gene/Q platform.
For the OpenMP+MPI runtime mode it was decided to use one MPI process per Blue Gene/Q node and 16 OpenMP threads on each node. This approach provides optimal resource utilization because of the Blue Gene/Q memory node allocation limitations and demonstrated good performance of the OpenMP threaded version of the model code. While each MPI application process allocates substantial amount of memory it is not possible to use maximal thread number per node without the code rearrangement. Preliminary scaling results have been shown in **Błąd! Nie można odnaleźć źródła odwołania.** with improvement in scaling of the mixed OpenMP+MPI threaded parallel version of the model at the beginning of the project (version **`2.8-v2`**) and at the project ending

(version `2.8-green`). With the improved application code linear scaling for up to 64 nodes is preserved. For larger numbers of MPI processes and higher node numbers problems with the I/O subsystem have been encountered. Further effort in enabling for BlueGene/Q specific I/O restrictions is needed to achieve full application scaling for this architecture.

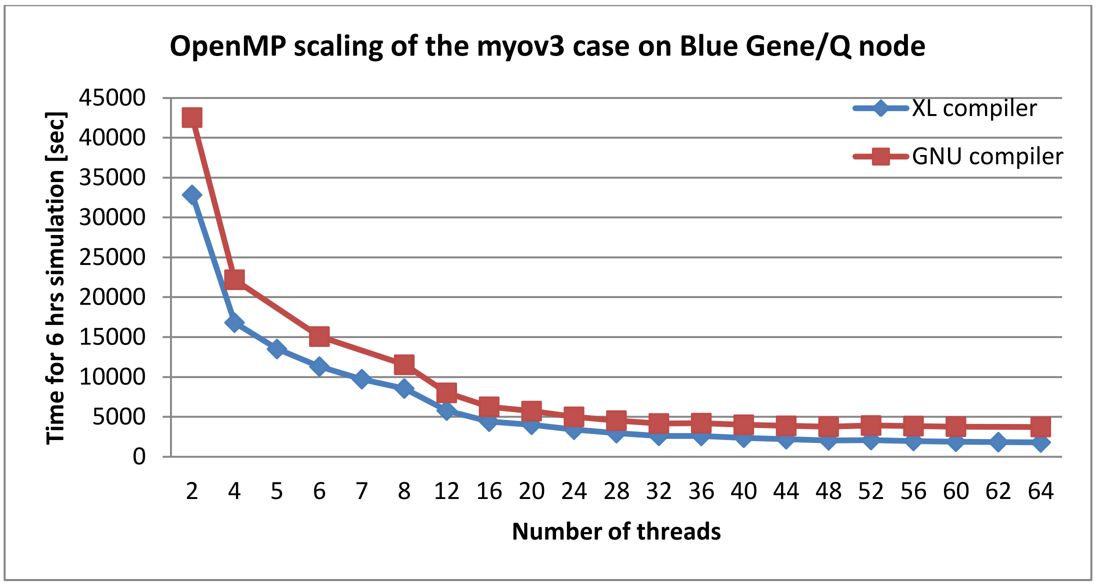

Figure 1: Thread scaling of the HBM model 2.8-v2, OpenMP version of the application tested with myov3 case on the one node of BlueGene/Q. Performance comparison between XL Fortran and GNU Fortran based applications.

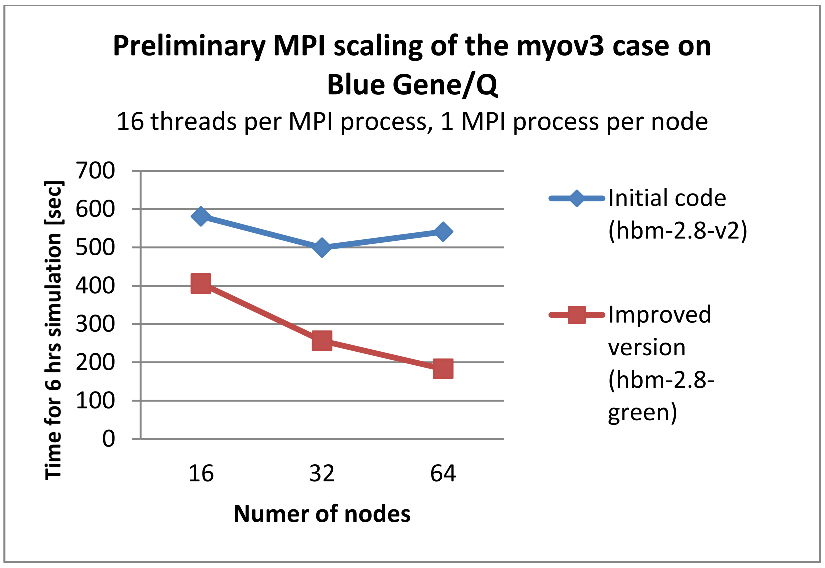

Figure 2: Preliminary results of the mixed threaded parallel model scaling

6. Conclusions and Outlook

Thread scaling on the BlueGene/Q architecture and micro-benchmarking approach for the HBM model have been described. The application code has been ported to the BlueGene/Q system and both threaded and mixed threaded parallel versions scaling have been tested. Optimal scaling for threaded application code has been reported including BlueGene/Q architecture specific features. Initial scaling tests with improved code version for threaded parallel (MPI+OpenMP) mode show also optimal scaling for small node counts.
To enable further scalability for the BlueGene/Q system two identified issues need to be addressed. Memory management for mixed threaded parallel mode and I/O management with a larger number of processes. Micro-

benchmarking procedure also needs to be integrated into the model code. These need to be regarded as a recommendations for further code improvements within the application development process.

References

[1] Jacob Weismann Poulsen and Per Berg, Thread scaling with HBM, Technical Report 12-20, Danish Meteorological Institute, http://beta.dmi.dk/fileadmin/user_upload/Rapporter/tr12-20.pdf

[2] Per Berg and Jacob Weismann Poulsen, Implementation details for HBM, Technical Report 12-11, Danish Meteorological Institute, http://beta.dmi.dk/fileadmin/Rapporter/TR/tr12-11.pdf

[3] Jacob Weismann Poulsen and Per Berg, More details on HBM - general modelling theory and survey of recent studies, Technical Report 12-16, Danish Meteorological Institute, http://beta.dmi.dk/fileadmin/Rapporter/TR/tr12-16.pdf

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement nos. RI-283493 and RI-312763.