Geoscientific Model Development Discussions

Geosci. Model Dev. Discuss., author comment AC1
https://doi.org/10.5194/gmd-2021-319-AC1, 2022

Reply on RC1

Giorgio Micaletto et al.

Author comment on "Parallel implementation of the SHYFEM (System of HydrodYnamic Finite Element Modules) model" by Giorgio Micaletto et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2021-319-AC1, 2022

Dear Reviewer

Thank you for your valuable comments. Here we report your comments followed by our answers in italic

As claimed by the authors the limiting factors to the overall scalability lie in the sea level computation which involves the PETSc – KSP solver and data decomposition. The efficiency drops below 55% with 144 cores. In particular,
Matrix decomposition used in PETSc, namely the block row partition, is not the same of SHYFEM. Then, a global communication is required and a loss of efficiency results.
My questions: are
since the parallelism is being introduced in the original sequential version of SHYFEM, why the data partitioning chosen for SHYFEM is not the same of PETSC?

The grid points (nodes) are ordered after application of the Cuthill-McKee algorithm on the mesh. This approach is used in continuity with the sequential version of SHYFEM, where the Cuthill-McKee algorithm is embedded in the SPARSEKIT framework. This is done to optimize the bandwidth of the sea level matrix and make the iterative solution faster. PETSc uses a 2D domain decomposition based on the ordering of the nodes in the matrix without considering the geometry; whilest, the grid partitioning used in SHYFEM takes into account the load balancing for the computation made on the 3D domain and to minimize the spatial cutting edge between one process and its neighbors (regardless of the numbering of the elements in the matrix). Moreover, the decomposition in SHYFEM is done considering the elements and not nodes as in PETSc. The decomposition used in SHYFEM is optimal to reduce the communications overhead and to better balance the workload in all other aspects of the model which computationally represent the most onerous part.
However, we are going to investigate the use of the DMPlex module available within PETSc to handle unstructured grids.

PETSc offers the Distributed Arrays (DMDA) objects that simplify the distribution and the management of the domain data (all the physical quantities on the domain region) in a distributed memory system. Why the authors do not exploit DMDA objects?

Thanks for the suggestion. We have investigated the usage of DMDA. To our knowledge, the DMDA objects are exclusively oriented to structured grids, as stated from PETSc

manual: "structured grid in 1, 2, or 3 dimensions. In the global representation of the vector each process stores a non-overlapping rectangular". In our case the SHYFEM model uses an unstructured grid.

Otherwise, why do not explore the use of TRILINOS, or HYPRE that are already able to interact with PETSc ?

We thank the reviewer for pointing out this aspect. The exploitation of Trilinos and Hypre libraries is under evaluation and we plan to include them in the code as soon as we have evidence of an improvement of the computational performance w.r.t. the current implementation, unfortunately this activity takes much more time than expected and cannot be considered for this manuscript. Even if the current implementation is perfectible, we believe we have reached a milestone by parallelizing the SHYFEM model with MPI and introducing the use of high performance numerical libraries.

Finally, why the authors do not explore matrix-free solvers?

We thank the reviewer for this suggestion, we have implemented a matrix-free approach and compared the computational performance w.r.t. the current implementation. The new approach did not bring an evident improvement, we will report this analysis in Section 5.2 of the paper. In particular, the matrix free approach requires the user to explicitly write the routine for the matrix-vector multiplication, however in order to correctly compute the matrix-vector multiplication a communication stage among the processes is needed. This communication time partially invalidates the benefit given by the matrix-free approach.

The KSP solver used for the free surface equation is known to have synchronization points at each iteration leading to a loss of efficiency for the parallel algorithm. My question is: why do not explore communication avoiding variants of Krylov sparse solvers?

We tested some other solvers which exploit the communication avoiding technique. This further analysis will be reported in Section 5.2. We evaluated the following solvers: Generalized Minimal Residual(GMRES); Improved Biconjugate gradient stabilized method(IBCGS), Flexible Biconjugate gradient stabilized (FBCGSR) and Biconjugate gradient stabilized method(BCGS). Among these methods the most efficient one, in our configuration, was the FBCGSR. We further investigated two pipelined methods such as PIPEBCGS and PIPEFGMRES: the first method diverges after a few iterations, it was necessary to increase the tolerances by 4 orders of magnitude in order to use it and it did not lead to improvements, the second instead had a worse scalability than the BCGS method used initially.

In addition to these two factors, in my opinion there is another crucial point. Since domain decomposition involves only spatial direction and not the time direction, a global communication is required at each time step of surface equation. My question is:
why do not explore parallel -in -time approaches ? Parallelism should be introduced ab initio in any mathematical / numerical model, and this is especially true for time marching models. Otherwise, the efficiency will be ever poor.

We thank the reviewer for pointing out this aspect. The parallel in time approach would require a complete and disruptive revision of the data structure and the whole code structure.

In conclusion, I think that while the deployment of an application software by means of the use of scientific libraries, such as PETSc, can be considered a good investment, SHYFEM needs to be deeply redesigned to meet scalability requirements.

We are aware that the code can be further optimized in the future. Our work followed an

evolutionary approach starting from an already existing sequential implementation of the model. A deep optimization would require a complete redesign of the code structure but this was out of the scope of our work.